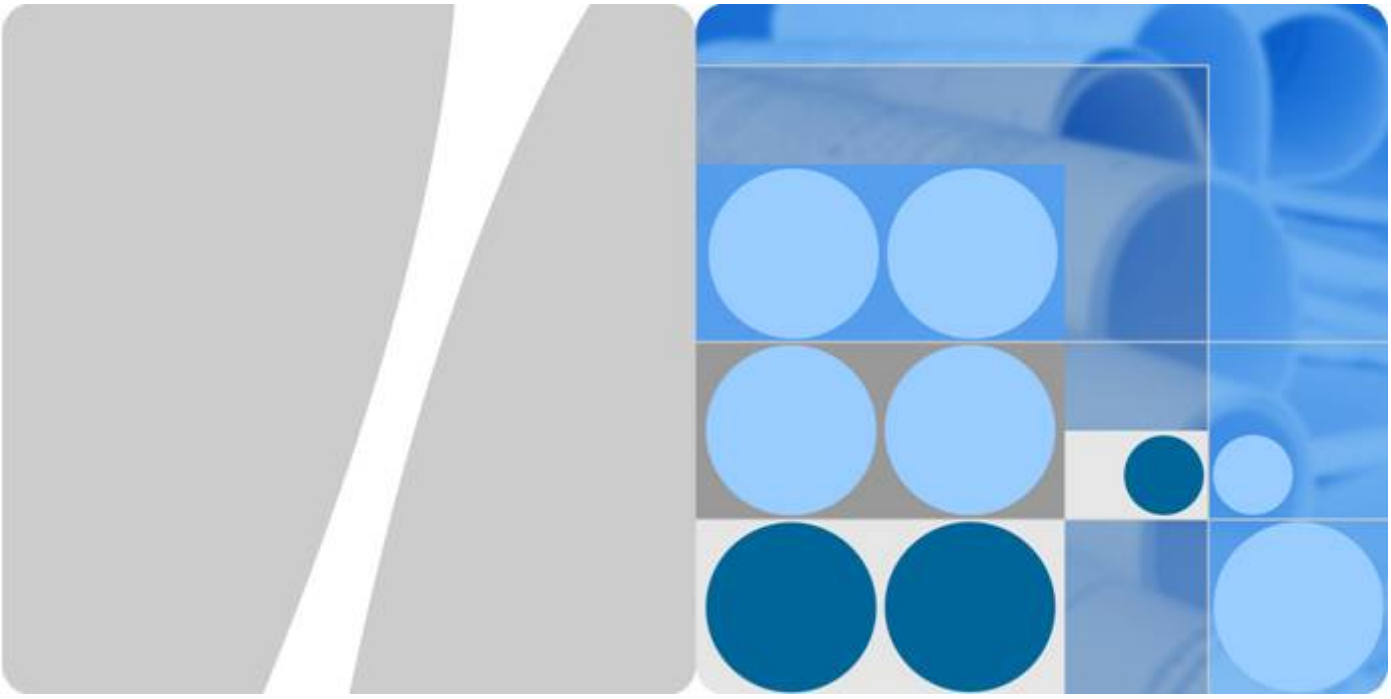


CodeArts PerfTest

User Guide

Date **2025-04-30**

Contents

1 Service Overview

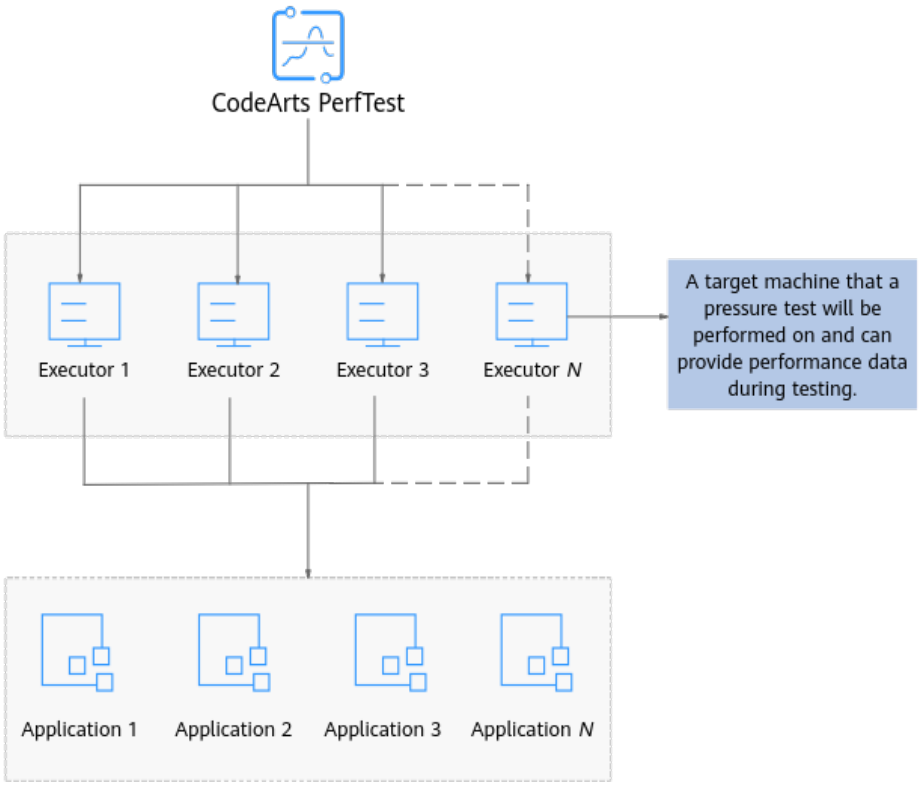
1.1 What Is CodeArts PerfTest?

Distributed architecture and microservice technologies have made applications more complex. This complexity results in architecture decoupling and performance improvements. However, it also makes it harder to locate performance problems in the production environment, and the repair periods become longer. Performance tests in advance of application launches are therefore necessary.

CodeArts PerfTest provides performance test services for cloud applications that are built based on HTTP, HTTPS, TCP, UDP, HLS, RTMP, WebSocket, MQTT, or HTTP-FLV. CodeArts PerfTest quickly simulates service peaks with a large number of concurrent users. It allows you to define the contents and time sequences of packets and supports complex combinations of multiple transactions. After tests are complete, CodeArts PerfTest provides professional test reports to evaluate your service quality.

CodeArts PerfTest simplifies performance pressure tests, helping you focus more on services and performance problems, reduce costs, enhance stability, optimize user experience, and improve the business value of enterprises.

Figure 1-1 CodeArts PerfTest



Functions

CodeArts PerfTest provides tests for HTTP/HTTPS/TCP/UDP/HLS/RTMP/WebSocket/HTTP-FLV/MQTT applications with high user concurrency. It allows you to flexibly define multi-protocol packet contents, transactions, and test task models. You can view performance statistics, such as concurrency, RPS, and response time during or after testing. You can also create private test clusters or scale in or out resource groups to test at different scales.

Multi-protocol and high-concurrency performance tests

- Quickly define standard HTTP/HTTPS/TCP/UDP/HLS/RTMP/WebSocket/HTTP-FLV/MQTT packet contents. You can send pressure test traffic to different tested applications through simple adjustments.

  Define any fields in HTTP/HTTPS/TCP/UDP/HLS/RTMP/WebSocket/HTTP-FLV/MQTT packets based on the requirements of tested applications. For example, you can configure methods such as GET/POST/PATCH/PUT/DELETE, and URLs, headers, and bodies of HTTP requests.
- Define the behavior of virtual users for different test scenarios.

  Specify the interval for sending requests of the same user by setting the think time, or define multiple request packets in a transaction to set the number of requests initiated by each user per second.
- Customize the validation for the response result to make the checkpoint for successful requests more accurate.

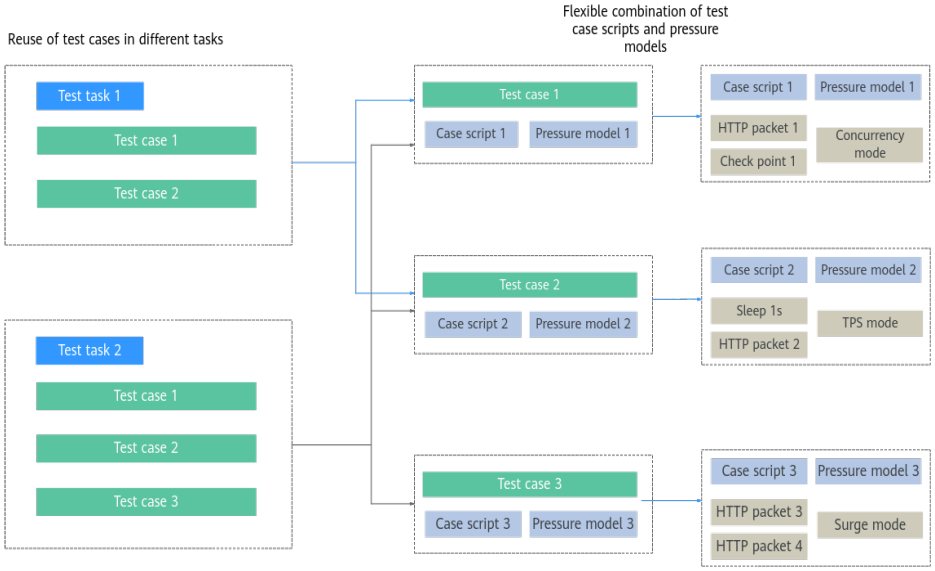
  CodeArts PerfTest allows you to configure checkpoints based on your service requests. After obtaining response packets, CodeArts PerfTest verifies their

response code, header and body fields. Only response packets that meet the specified conditions are considered normal responses.

Defining test task models for complex scenario tests

- You can test application performance by combining case scripts and pressure models for different high-concurrency scenarios.
- Test cases can be reused. You can define their pressure model and parameters such as the duration, number of concurrent users, number of flapping, and surge times, to simulate complex scenarios with traffic peaks and troughs.

Figure 1-2 Complex scenario tests



Providing professional performance test reports for easy understanding of application performance

- CodeArts PerfTest collects application performance statistics from multiple aspects, such as RPS, number of concurrent users, response latency, number of access requests, response verification failures, and response timeout.
- CodeArts PerfTest provides real-time and offline test reports for you to view and analyze test data at any time.

Managing private test clusters, isolating tenants' traffic, and providing on-demand use

- You can create test clusters as required to isolate the traffic between tenants and complete pressure tests of the internal network and external network. After tests are complete, you can delete clusters at any time.
- CodeArts PerfTest supports real-time scale-ins, scale-outs, and upgrades of test clusters.

1.2 Advantages

CodeArts PerfTest provides a one-stop performance test solution, helping you identify performance bottlenecks of applications in advance.

Cost-Efficient Simulation of Ultra-High Concurrency

- CodeArts PerfTest provides you with private test clusters. In such a test cluster, a single execution node can simulate tens of thousands of virtual users, and the entire test cluster can simulate hundreds of thousands virtual users.
- CodeArts PerfTest is easy to use and greatly reduces test time. It simulates hundreds of thousands instantaneous concurrent requests. In this way, you can identify application performance bottlenecks in high-concurrency scenarios and prevent system breakdown caused by a large number of access requests.
- CodeArts PerfTest supports execution of multiple concurrent tasks. It enables you to test the performance of multiple applications at the same time, greatly improving test efficiency.

Flexible and Fast Performance Testing, Achieving Quick Application Rollout

- Flexible protocol customization: HTTP/HTTPS tests are used to test the performance of various applications and microservice interfaces developed based on the HTTP/HTTPS protocol. TCP/UDP/WebSocket tests support the string and hexadecimal code stream modes, which meet the data construction requirements of various non-HTTP protocols. HLS/RTMP/HTTP-FLV/MQTT tests are supported.
- Flexible combination of multiple transaction elements and test task phases: CodeArts PerfTest provides flexible definition of data packets and transactions, as well as simulates scenarios where multiple users perform transaction operations during traffic peaks and troughs of test tasks. All of these features make CodeArts PerfTest ideal for complex scenario tests. In addition, CodeArts PerfTest allows you to specify the number of concurrent users for each transaction at each period and simulates instantaneous service traffic.

On-demand Use of Resources in Performance Tests

- Private resource group: You can create test clusters as required to isolate the traffic between tenants and complete pressure tests of the internal network and external network. After tests are complete, you can delete clusters at any time. CodeArts PerfTest supports real-time scale-ins, scale-outs, and upgrades of test clusters.
- Shared resource group: Use shared resource groups for debugging or small scale concurrent pressure tests.

Quick Location of Performance Bottlenecks

CodeArts PerfTest provides professional performance test reports to show metrics such as transaction concurrency, RPS, throughput, and response latency, metrics that worth attention in order to provide a pleasant use experience. CodeArts PerfTest provides real-time and offline reports, allowing for analysis of test data at any time.

1.3 Application Scenarios

CodeArts PerfTest provides distributed pressure tests and is widely used in various industries, such as the Internet, digital marketing, Internet of Vehicles (IoV), and finance.

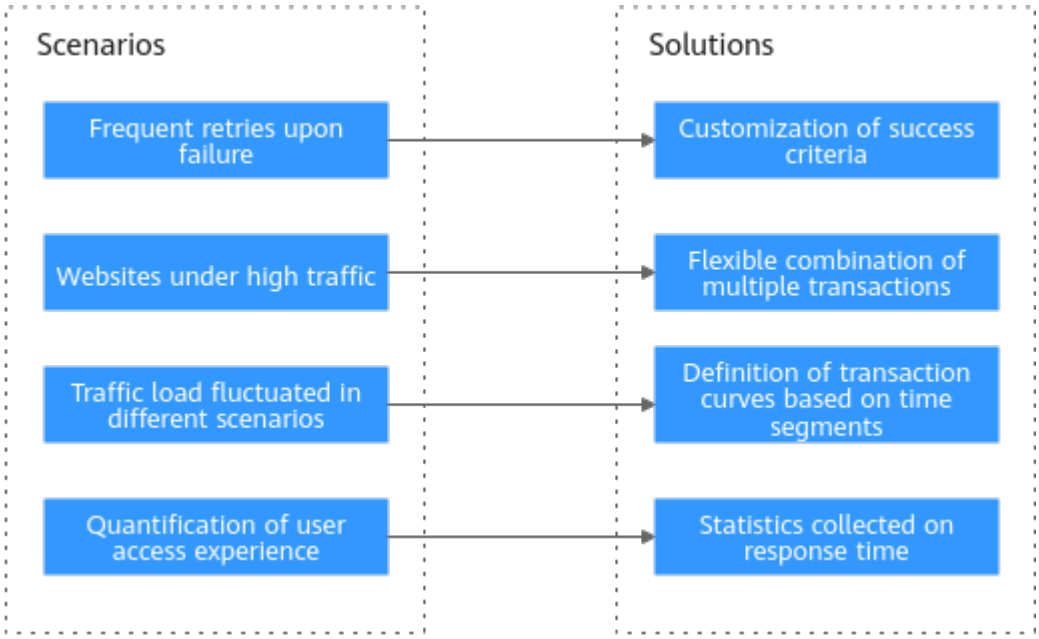
E-Commerce Flash Sale Tests

E-commerce flash sales are characterized by large-scale user concurrency, multiple burst requests, and repeated access attempts. Guaranteeing the availability of websites under heavy load is key.

Advantages

- Scenario simulation: CodeArts PerfTest simulates hundreds of thousands instantaneous concurrent requests to a website, and can simulate a heavy-load website in just one test model.
- Professional test report: CodeArts PerfTest provides statistics on the response latency range that accurately reflect user experiences.
- Retry for failed users: User-defined comparison of results calculated using expressions allows users who failed to log in to retry.

Figure 1-3 E-commerce flash sale tests



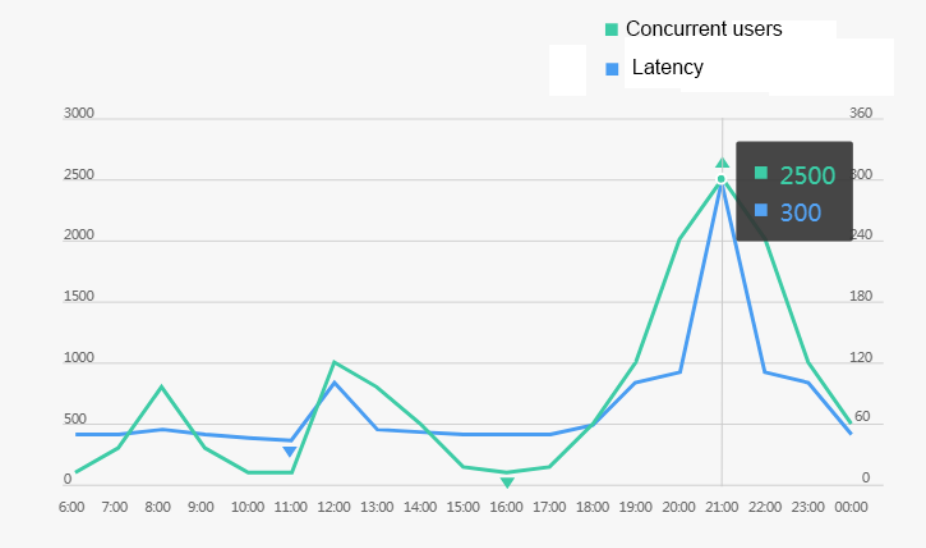
Game Peak Tests

Game services feature auto scaling in peak and off-peak scenarios. You can verify if auto scaling of games is normal, and if KPIs meet requirements in burst traffic scenarios.

Advantages

- Multi-scenario combination simulation: CodeArts PerfTest simulates real scenarios by combining multiple transactions, which include diverse elements, and customizing packets.
- Peak and off-peak scenario simulation: CodeArts PerfTest develops a pressure test curve for each transaction within a defined period to simulate peak and off-peak scenarios.
- KPI measurement: You can verify game KPIs in a peak scenario based on a customized response timeout interval.

Figure 1-4 Game peak tests



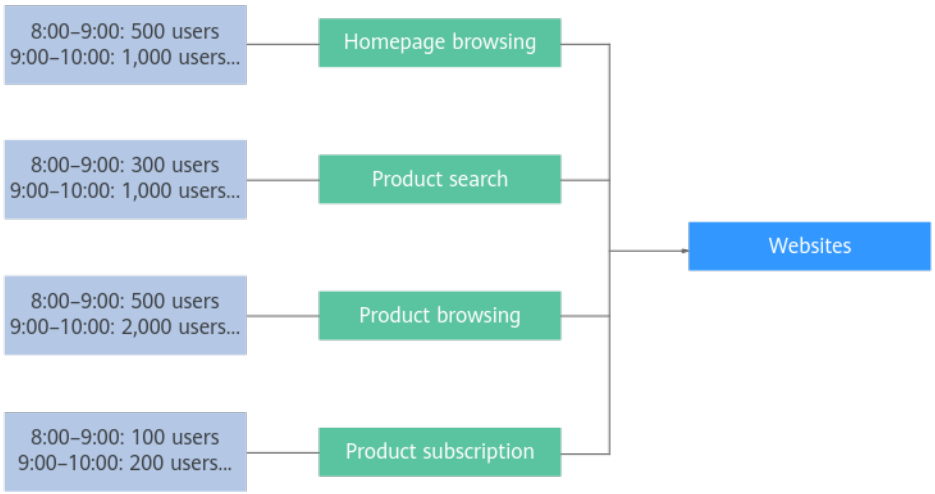
Complex Scenarios Support

CodeArts PerfTest simulates all the complexities of real traffic: A user's access may involve multiple HTTP requests; different users perform different transaction operations; user access fluctuates with transactions, displaying a sharp peak and off-peak trend; there may be instantaneous concurrent users. Performance tests must be performed on services to identify performance bottlenecks in advance.

Advantages

- Flexible model customization: CodeArts PerfTest supports multi-transaction tests in which multiple users perform multiple operations.
- Burst traffic: CodeArts PerfTest allows you to specify the number of concurrent users for each transaction at each period and simulates instantaneous service traffic.
- Result verification: Customized comparison of results calculated using expressions helps you customize transaction standards.

Figure 1-5 Complex scenarios support



1.4 Notes and Constraints

Test Resource Groups and Their Constraints

- Test resource groups are classified into shared resource groups and private resource groups. Shared resource groups are provided by the system by default, and private resource groups need to be created.
- Execution nodes of the shared resource group have been bound with an elastic IP address (EIP). When the tested application has network access restrictions, use a private resource group.
- In the same region, a shared resource group supports PerfTest test tasks with up to 10,000 or 100,000 concurrent users in total (determined by the package you selected when installing CodeArts PerfTest). If the number of concurrent users exceeds the upper limit, use a private resource group.
- JMeter test tasks can use only private resource groups.

Suggestions on Using Nodes

- If an application is deployed on a node in a cluster, the node cannot be selected to create a private resource group. Do not run any applications or perform other functions on nodes used for test resource groups. Otherwise, applications may run abnormally.
- If you want to perform pressure tests on external services, bind an EIP to each execution node. If you want to debug external services, bind EIPs to both the debugging node and execution node. The test bandwidth is limited by the EIPs' bandwidth.
- Create at least three empty nodes. Two are for debugging an execution node. The other is the execution node/executor (a target machine that a pressure test will be performed on and can provide performance data during testing). Create nodes of the required specifications based on the number of concurrent users for a pressure test. For details about the recommended node specifications, see **Table 1-1** and **Table 1-2**. These specifications are for reference only. Resource specification requirements for a pressure test are influenced by think time, protocol type, the size and number of requests and responses, response time, and result verification. Adjust the specifications as needed.
- In a PerfTest test project, one execution node with 8 vCPUs and 16 GB memory supports 10,000 concurrent users. In a JMeter test project, one execution node with 8 vCPUs and 16 GB memory supports 2,000 concurrent users.

Table 1-1 Recommended node specifications for PerfTest projects

Concurrent Users	Specifications	Quantity
0–5,000	Debugging node: 4 vCPUs | 8 GB	2

Concurrent Users	Specifications	Quantity
	Execution node: 4 vCPUs | 8 GB	1
5,001–10,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	1
10,001–20,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	2
20,001–30,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	3
30,001–40,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	4
40,001–50,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	5
More than 50,001	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	*n*

Table 1-2 Recommended node specifications for JMeter projects

Concurrent Users	Specifications	Quantity
0–1,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 4 vCPUs | 8 GB	1
1,001–2,000	Debugging node: 4 vCPUs | 8 GB	2

Concurrent Users	Specifications	Quantity
	Execution node: 8 vCPUs | 16 GB	1
2,001–4,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	2
4,001–6,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	3
6,001–8,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	4
8,001–10,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	5
More than 10,001	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	*n*

1.5 Quotas

Pay attention to the quota limits listed in **Table 1-3**.

Table 1-3 Quota limits

Item	Limit	Scope
Number of transactions and JMeter test plans	200	Tenant
Number of PerfTest and JMeter projects (total)	100	Tenant
Number of PerfTest and JMeter file variables (total)	100	Tenant
Number of crontasks	20	Tenant

Item	Limit	Scope
Number of third-party JMeter JAR packages	10	Tenant
Number of tasks	200	Project
Number of global variables	100	Project
Number of created test cases	2,000	Project
Number of added test cases	30	Task
Size of a file in JMeter test plans	10 MB	Region
Size of a CSV file in PerfTest or JMeter variable files	80 MB	Region
Size of an XLSX file in PerfTest or JMeter variable files	10 MB	Region
Number of concurrent tasks	Based on the maximum concurrency you selected when installing CodeArts PerfTest	Region

1.6 Basic Concepts

Execution Node

An execution node is the target machine that a pressure test will be performed on and can provide performance data during testing.

Debugging Node

A debugging node is used to debug execution nodes.

Test Resources

Test resources refer to private resource groups.

CodeArts PerfTest Resources

CodeArts PerfTest resources refer to items such as test projects, directories, cases, and tasks.

Test Project

Test projects are classified into PerfTest projects and JMeter projects. PerfTest projects provide project management capabilities, allowing you to share and reuse the contents of transactions, pressure test tasks, and test reports within a test project, and create different test projects for different test programs. JMeter projects are used to import JMeter scripts to CodeArts PerfTest.

Transaction

A transaction is a user-defined operation model that consists of HTTP/HTTPS/TCP/UDP/WebSocket packet, think time, response extraction, checkpoint, and HLS/RTMP/HTTP-FLV/MQTT packet.

Packet

Packets are data blocks transmitted between applications such as HTTP. These data blocks start with text metadata that describes the packet content and meaning. The text metadata is followed by optional data. Packets are transmitted among clients, servers, and agents.

Think Time

To better simulate user behavior, insert a waiting time between different operations. For example, when a user receives data from the server, the user may wait several seconds before viewing data and providing responses. This period of time is called think time.

Response Extraction

If a transaction contains multiple packets, the output of the previous packet, which is extracted by a regular expression or JSON, is used as the input of the next packet.

Checkpoint

Checkpoints are where you define the verification information to determine whether the contents returned by the server are correct.

Test Task

A test task initiates a performance test based on a defined test model.

Test Report

When a test task is complete, a test report will be generated to present the test results.

Number of Concurrent Users

It refers to the number of users performing operations on a system at the same time. In CodeArts PerfTest, it is the number of virtual users you define when you configure a test phase.

RPS

Requests per second (RPS) indicates the number of requests per second. Average RPS = Total number of requests in a statistical period/Statistical period.

VUM

Virtual user minute (VUM) indicates the number of resources consumed by a task. The calculation formula is VUM = VU x M, in which VU indicates the number of concurrent virtual users and M indicates the pressure test duration, in minutes.

Bandwidth

Bandwidth records the real-time bandwidth usage during the running of the pressure test task. Uplink bandwidth refers to the speed at which the CodeArts PerfTest execution node sends out data. Downlink bandwidth refers to the speed at which the CodeArts PerfTest execution node receives data.

Response Time

Response time indicates the duration from the time when a client sends a request to the time when the client receives a response from the server.

Response Timeout

If the corresponding TCP connection does not return the response data within the set response timeout (5 seconds by default), the transaction request is considered a response timeout. Possible causes are: the tested server is busy, in crashes, or the network bandwidth is fully occupied.

Verification Failure

The response packet content and response code returned from the server do not meet the expectation (the default expected response code of HTTP/HTTPS is 200), such as code 404 or 502. A possible cause is that the tested service cannot be processed normally in scenarios with a large number of concurrent users. For example, a database bottleneck occurs in the distributed system or the backend application returns an error.

Resolution Failure

All response packets are received, but some packets are lost. As a result, the entire transaction response is incomplete. In this case, network packet loss may be the cause.

1.7 Permissions

CodeArts PerfTest Permissions

New IAM users do not have any permissions assigned by default. You need to first add them to one or more groups and then attach policies or roles to these groups.

The users then inherit permissions from the groups and can perform specified operations on cloud services based on the permissions they have been assigned.

You can grant permissions by using roles and policies. Currently, only authorization by roles is supported in CodeArts PerfTest.

- Roles: A coarse-grained authorization strategy that defines permissions by job responsibility. Only a limited number of service-level roles are available for authorization. Cloud services often depend on each other. When you grant permissions using roles, you also need to attach any existing role dependencies. Roles are not ideal for fine-grained authorization and least privilege access.
- Policies: A fine-grained authorization strategy that defines permissions required to perform operations on specific cloud resources under certain conditions. This type of authorization is more flexible and is ideal for least privilege access.

Table 1-4 lists all the system-defined permissions for CodeArts PerfTest.

Table 1-4 System-defined permissions for CodeArts PerfTest

Role/ Policy Name	Description	Details	Type	Dependencies
CodeArts PerfTest Administrator	Users with these permissions have full permissions for CodeArts PerfTest.	Users with these permissions can perform all operations on CodeArts PerfTest and test resources of the current tenant and all IAM users, such as adding, deleting, modifying, and querying resources.	System-defined role	To create, modify, or delete private resource groups, the **CCE Administrator**, **Server Administrator**, and **VPC Endpoint Administrator** permissions need to be assigned to users. If the user uses only the shared resource group to perform operations, no other permissions are required.

Role/ Policy Name	Description	Details	Type	Dependencies
CodeArts PerfTest Developer	Users with these permissions have full permissions for their own resources, but have no permission for those of other users under the tenant.	Users with these permissions can perform all operations, such as adding, deleting, modifying, and querying resources, only on a user's own CodeArts PerfTest and test resources.	System-defined role	To create, modify, or delete private resource groups, the **CCE Administrator**, **Server Administrator**, and **VPC Endpoint Administrator** permissions need to be assigned to users. If the user uses only the shared resource group to perform operations, no other permissions are required.
CodeArts PerfTest Operator	Users with these permissions have read-only permissions for CodeArts PerfTest.	Users with these permissions can only read their own CodeArts PerfTest and test resources.	System-defined role	None.
CodeArts PerfTest Resource Administrator	Users with these permissions have all permissions related to test resources in CodeArts PerfTest.	Users with these permissions have all permissions related to test resources in CodeArts PerfTest.	System-defined role	This role must be used together with the **CodeArts PerfTest Developer** role, so that **CodeArts PerfTest Developer** can add, delete, modify, and query all private resource groups under the tenant.
CodeArts PerfTest Resource Developer	Users with these permissions can only view and use CodeArts PerfTest resources, but cannot create, update, or delete infrastructure resources.	Users with these permissions can only view and use CodeArts PerfTest resources, but cannot create, update, or delete infrastructure resources.	System-defined role	This role must be used together with the **CodeArts PerfTest Developer** role, so that **CodeArts PerfTest Developer** can view and use all private resource groups under the tenant.

Notes and Constraints

- To manage private resource groups, you must have the **CodeArts PerfTest Administrator**, **Server Administrator**, **CCE Administrator Role**, and **VPC Endpoint Administrator** permissions.
- To use only private resource groups, you must have the **CodeArts PerfTest Administrator** and **VPC Endpoint Administrator** permissions.
- To use only shared resource groups, you must have the **CodeArts PerfTest Administrator** permission.

2 Quick Start

2.1 Introduction

CodeArts PerfTest provides performance test services for cloud applications that are built based on HTTP, HTTPS, TCP, UDP, HLS, RTMP, WebSocket, MQTT, or HTTP-FLV. CodeArts PerfTest quickly simulates service peaks with a large number of concurrent users. It allows you to define the contents and time sequences of packets and supports complex combinations of multiple transactions. After tests are complete, CodeArts PerfTest provides professional test reports to evaluate your service quality.

You can complete a performance test in four steps listed as in the following figure.

Figure 2-1 Performance test procedure

You can quickly get familiar with the CodeArts PerfTest usage through the **interactive walkthroughs**.

Basic Concepts

- **Test Project:** Test projects are classified into PerfTest projects and JMeter projects. PerfTest projects provide project management capabilities, allowing you to share and reuse the contents of transactions, pressure test tasks, and test reports within a test project, and create different test projects for different test programs. JMeter projects are used to import JMeter scripts to CodeArts PerfTest.
- **Transaction:** A transaction is a user-defined operation model that consists of HTTP/HTTPS/TCP/UDP/WebSocket packet, think time, response extraction, checkpoint, and HLS/RTMP/HTTP-FLV/MQTT packet.
- **Packet:** Packets are data blocks transmitted between applications such as HTTP. These data blocks start with text metadata that describes the packet content and meaning. The text metadata is followed by optional data. Packets are transmitted among clients, servers, and agents.

- **Think Time:** To better simulate user behavior, insert a waiting time between different operations. For example, when a user receives data from the server, the user may wait several seconds before viewing data and providing responses. This period of time is called think time.
- **Response Extraction:** If a transaction contains multiple packets, the output of the previous packet, which is extracted by a regular expression or JSON, is used as the input of the next packet.
- **Checkpoint:** Checkpoints are where you define the verification information to determine whether the contents returned by the server are correct.
- **Number of Concurrent Users:** It refers to the number of users performing operations on a system at the same time. In CodeArts PerfTest, it is the number of virtual users you define when you configure a test phase.
- **Response Time:** Response time indicates the duration from the time when a client sends a request to the time when the client receives a response from the server.

2.2 Interactive Walkthroughs

Scenarios

You can quickly get familiar with the CodeArts PerfTest usage through the interactive walkthroughs.

The interactive walkthroughs provide the following four experience wizards:

- Using the testing service: helps you get familiar with basic CodeArts PerfTest operations, including deploying a test project, adding a test task, and generating a test report.
- Variable debugging: guides you how to add global and local variables, quickly define a pressure-testing model, and check whether the configuration is correct when debugging.
- E-commerce solution: supports full-link pressure testing in scenarios with large numbers of concurrent users and multiple transactions. This wizard also guides you how to test E-commerce websites and solve problems such as application breakdown and capacity expansion.
- All-in-one system template: a pressure test template built for all-in-one systems, featuring ease of use, scenario simulation, and various pressure configurations.

Procedure

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **Interactive Walkthroughs**.

Step 3 Choose a wizard and click **Try Now**.

Complete the experience as prompted.

----End

2.3 Preparing Environment Resources

Test Resource Groups and Their Constraints

- Test resource groups are classified into shared resource groups and private resource groups. Shared resource groups are provided by the system by default, and private resource groups need to be created.
- Execution nodes of the shared resource group have been bound with an elastic IP address (EIP). When the tested application has network access restrictions, use a private resource group.
- In the same region, a shared resource group supports PerfTest test tasks with up to 10,000 or 100,000 concurrent users in total (determined by the package you selected when installing CodeArts PerfTest). If the number of concurrent users exceeds the upper limit, use a private resource group.
- JMeter test tasks can use only private resource groups.

Creating a Private Resource Group

Step 1 Log in to the CodeArts PerfTest console, choose **Resource Groups** in the navigation pane, and click **Create Resource Group**.

Step 2 (Optional) If this is the first time you create a private resource group, grant CodeArts PerfTest permissions necessary for creating private resource groups.

Step 3 If no CCE cluster is available, **create a cluster** and then **create a resource group**. If a CCE cluster is available, **create a resource group**.

Step 4 Create a cluster.

Click **Create Cluster**. The page for creating CCE clusters is displayed. For details about how to create a cluster, see Help Center > Cloud Container Engine > User Guide > "Clusters" > "Creating a CCE Cluster".

Configure the cluster parameters as follows:

- The cluster management scale is related to the number of execution nodes. Create nodes of the corresponding specifications based on the number of concurrent users for pressure testing. For example, if 20 execution nodes are required, you can set the cluster scale to 50 nodes.
- You are advised to select **Tunnel network** as the network model. **Container CIDR Block** and **Service CIDR Block** must be the same as those of the tested object.

Click **Next: Select Add-on**. On the page displayed, select the add-ons to be installed during cluster creation. When selecting add-ons, retain the default settings for the test executor. For example, deselect unnecessary add-ons, such as NodeLocal DNSCache and Cloud Native Cluster Monitoring, to prevent the add-ons from occupying executor resources.

Click **Next: Add-on Configuration**. Retain the default settings.

Click **Next: Confirm configuration**. After confirming that the cluster configuration is correct, select **I have read and understood the preceding instructions** and click **Submit**. It takes about 6 to 10 minutes to create a cluster.

After the cluster is created, return to the cluster management page and click **Create Node**. For details about how to create a node, see Help Center > Cloud Container Engine > User Guide > "Nodes" > "Creating a Node".

Configure the node parameters as follows:

- A node must have at least 4 vCPUs and 8 GB memory.
- Select EulerOS as the operating system.
- At least three nodes (two debugging nodes and one execution node) are required. The number of nodes depends on the specifications of the pressure test object. For example, 22 execution nodes (two debugging nodes and 20 execution nodes) are required for 100,000 concurrent users and 4 vCPUs | 8 GB memory.
- If the CCE cluster node and the application to be tested are not in the same VPC network, bind an EIP to the CCE cluster node. You can use existing EIPs. If no EIP is available, click **Auto create** to create one. When EIPs are automatically created, you are advised to set the bandwidth to a large value to avoid affecting the pressure test. Additionally, an EIP with specified configurations is automatically assigned to each node. If the number of EIPs is less than the number of nodes, the EIPs are randomly bound to the nodes.
- In the **(Optional) Advanced Settings** area, set **Kubernetes Node Name** to **Node private IP**. If you select **Cloud server name**, the node cannot be managed.

Click **Next: Confirm**. After confirming that the node configuration is correct, select **I have read and understood the preceding instructions** and click **Submit**. After the node is created, return to the CodeArts PerfTest console.

Step 5 Create a resource group.

Choose **Resource Groups** in the navigation pane, and click **Create Resource Group**.

Set the parameters listed in **Table 2-1**.

Table 2-1 Creating a private resource group

Parameter	Description
Resource Group Name	Name of the private resource group, for example, **Web-test-demo**
Node Cluster	Select a CCE cluster from the drop-down list.
Debugging Node	Node used for debugging. It cannot be changed after the resource group is created. At least two debugging nodes need to be created.
Execution Node	Target machine that a pressure test will be performed on and can provide performance data during testing

Click **Create**.

----End

2.4 Creating a Test Project

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **PerfTest Projects** in the navigation pane, and click **Create Test Project**.

Step 2 In the displayed dialog box, enter a test project name (for example, **Web-test**) and description, and click **OK**.

----End

2.5 Creating a Test Case

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the navigation pane.

Step 2 Locate the row that contains the target PerfTest project, and click the test project name, for example, **Web-test**. The test project details page is displayed. On the **Cases** tab, you can view the default directory and sample cases that are automatically generated.

Step 3 In the default directory, choose **Sample Case**. Click **Case Script** and select **Pre-Steps** or **Steps**.

Pre-Steps are executed only once in each execution thread. This configuration is optional.

Steps are executor of a test case, which needs to be configured.

Step 4 In the case, click **Add Request**. Enter a request name and add requests.

- Packet

  (Mandatory) Packets are data blocks sent between HTTP applications.

  Click the **Packet** tab, and set packet parameters based on the actual service to be tested.

- Think time

  (Optional) To better simulate user behavior, it is necessary to simulate the waiting time between different operations. For example, when you receive data from a server, you may wait for several seconds to view the data before responding. This latency is called **Think Time**.

  Set **Think Time** based on requirements. For more effective testing, do not set a long think time. You are advised to test the server for the worst case scenario.

  Enable **Think Time** and set **Duration** to **1000** ms.

- Response extraction

  (Optional) If multiple packets exist in the same case, use regular expressions or JSON to extract the output of the previous packet for the input of the next packet.

Enable **Response Extraction** and set parameters.

- Checkpoint

  (Optional) When enabled, you can check whether the content returned by a server is correct through customized verification information.

  Enable **Checkpoint** and set parameters.

Step 5 Select **Pressure Stage**.

You can select different pressure modes and add multiple phases. Each phase can simulate different numbers of concurrent users.

Step 6 Click **Save**.

Step 7 Click **Debug** in the upper right corner of the page, select the target test resource group as the executor, and click **Start**.

Step 8 Click **View log** to view the test case debugging details.

If an error was reported in the debugging result, edit the case based on the log information and debug it again.

Step 9 On the **Debug log** tab page, you can view the debugging history.

----End

2.6 Creating a Test Task

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the navigation pane.

Step 2 Locate the row that contains the target PerfTest project, and click the test project name, for example, **Web-test**. The test project details page is displayed.

Step 3 On the **Test Tasks** tab, click **Create Task** on the upper right of the page.

Step 4 Enter the task name, for example, **taskA**, and select an execution mode.

There are two execution policies:

- **Serial**: Test cases in the test task are executed in sequence.
- **Parallel**: Test cases in the test task are executed concurrently.

Step 5 Click **Add Case**. In the displayed dialog box, select the created cases and click **OK**.

Step 6 Click **Save**.

----End

2.7 Viewing a Test Report

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the navigation pane.

Step 2 Locate the row that contains the target PerfTest project, and click the test project name, for example, **Web-test**. The test project details page is displayed.

Step 3 On the **Test Tasks** tab, select a test task, for example, **taskA**, and click ▶ in the **Operation** column.

Step 4 Select a resource group type and click **Start**.

Step 5 Click **View Report** to go to the real-time report page.

You can also click  in the **Operation** column of **taskA** after the pressure test task is complete to view the offline report.

On the **Overview** page, click **Download Report** in the upper right corner to obtain a PDF file. Contact experts to analyze the performance bottleneck of the current system and provide improvement suggestions.

----End

3 User Guide

3.1 CodeArts PerfTest Use Process

CodeArts PerfTest provides performance test services for cloud applications that are built based on HTTP, HTTPS, TCP, UDP, HLS, RTMP, WebSocket, MQTT, or HTTP-FLV. CodeArts PerfTest quickly simulates service peaks with a large number of concurrent users. It allows you to define the contents and time sequences of packets and supports complex combinations of multiple transactions. After tests are complete, CodeArts PerfTest provides professional test reports to evaluate your service quality.

You can complete a performance test in four steps listed as in the following table.

Table 3-1 Process

1. Prepare a Resource Group	2. Create a Test Project	3. Create a Test Task	4. View a Test Report
Prepare a test resource group for running the performance test. **NOTE** Test resource groups are classified into shared resource groups and private resource groups. Shared resource groups are provided by the system by default, and private resource groups need to be created.	CodeArts PerfTest manages your test projects. Transactions, test tasks, and test reports are shared in the same project.	Use predefined test transactions to create performance test scenarios.	CodeArts PerfTest provides real-time and offline performance test reports for result analysis.

Constraint

To use CodeArts PerfTest for a pressure test on a public website, the user must be on the whitelist of this website. Otherwise, the user must bear any and all consequences and legal liabilities.

Logging In to the CodeArts PerfTest Console

Step 1 Use a browser to log in to the management console.

URL: **https://***Address of the cloud service management console*, for example, **https://console.demo.com/**.

Step 2 In the service list, choose **CodeArts PerfTest**.

----End

3.2 Permissions Management

3.2.1 Creating a User and Assigning CodeArts PerfTest Permissions

This section describes how to use IAM to implement fine-grained permissions control for your CodeArts PerfTest resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing CodeArts PerfTest resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust other accounts or cloud services to perform efficient O&M on your CodeArts PerfTest resources.

If your account does not require individual IAM users, skip this chapter.

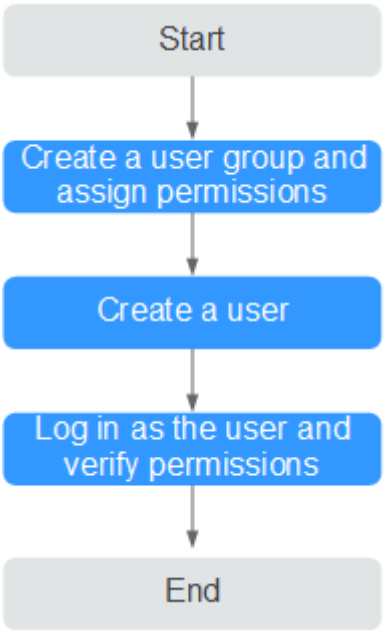
This section describes the procedure for granting permissions (see **Figure 3-1**).

Prerequisites

Before assigning permissions to user groups, you should learn about the system permissions listed in **Permissions**.

Example

Figure 3-1 Process of granting CodeArts PerfTest permissions



1. Create a user group and assign permissions to it.

   Create a user group on the IAM console and assign the **Administrator** permissions to the group.

   Create a user group on the IAM console and click **Authorize** in the **Operation** column to assign the **CodeArts PerfTest Administrator** permissions to the group.
2. Create an IAM user.

   Create a user on the IAM console and add it to the user group created in **1** by choosing **Authorize** in the **Operation** column.
3. Log in and verify permissions.

   Log in to the CodeArts PerfTest console as the created user, switch to the authorized region, and verify the permissions. Choose **Service List** > **CodeArts PerfTest**. On the CodeArts PerfTest console, choose **PerfTest Projects** in the navigation pane, and click **Create Test Project** on the right. If a test project can be created, the permissions are successfully configured.

3.3 Test Resource Management

3.3.1 Creating a Private Resource Group

Test Resource Groups and Their Constraints

- Test resource groups are classified into shared resource groups and private resource groups. Shared resource groups are provided by the system by default, and private resource groups need to be created.

- Execution nodes of the shared resource group have been bound with an elastic IP address (EIP). When the tested application has network access restrictions, use a private resource group.
- In the same region, a shared resource group supports PerfTest test tasks with up to 10,000 or 100,000 concurrent users in total (determined by the package you selected when installing CodeArts PerfTest). If the number of concurrent users exceeds the upper limit, use a private resource group.
- JMeter test tasks can use only private resource groups.

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **Resource Groups** in the navigation pane, and click **Create Resource Group**.

Step 2 (Optional) If this is the first time you create a private resource group, grant CodeArts PerfTest permissions necessary for creating private resource groups.

Step 3 If you do not have a CCE cluster, **create a cluster** and then **create a resource group**. If a CCE cluster is available, **create a resource group**.

Step 4 Create a cluster.

Click **Create Cluster**. The page for creating CCE clusters is displayed. For details about how to create a cluster, see Help Center > Cloud Container Engine > User Guide > "Clusters" > "Creating a CCE Cluster".

Configure the cluster parameters as follows:

- The cluster management scale is related to the number of execution nodes. Create nodes of the corresponding specifications based on the number of concurrent users for pressure testing. For example, if 20 execution nodes are required, you can set the cluster scale to 50 nodes.
- You are advised to select **Tunnel network** as the network model. **Container CIDR Block** and **Service CIDR Block** must be the same as those of the tested object.

Click **Next: Select Add-on**. On the page displayed, select the add-ons to be installed during cluster creation. When selecting add-ons, retain the default settings for the test executor. For example, deselect unnecessary add-ons, such as NodeLocal DNSCache and Cloud Native Cluster Monitoring, to prevent the add-ons from occupying executor resources.

Click **Next: Add-on Configuration**. Retain the default settings.

Click **Next: Confirm configuration**. After confirming that the cluster configuration is correct, select **I have read and understood the preceding instructions** and click **Submit**. It takes about 6 to 10 minutes to create a cluster.

After the cluster is created, return to the cluster management page and click **Create Node**. For details about how to create a node, see Help Center > Cloud Container Engine > User Guide > "Nodes" > "Creating a Node".

Configure the node parameters as follows:

- A node must have at least 4 vCPUs and 8 GB memory.
- Select EulerOS as the operating system.

- At least three nodes (two debugging nodes and one execution node) are required. The number of nodes depends on the specifications of the pressure test object. For example, 22 execution nodes (two debugging nodes and 20 execution nodes) are required for 100,000 concurrent users and 4 vCPUs | 8 GB memory.
- If the CCE cluster node and the application to be tested are not in the same VPC network, bind an EIP to the CCE cluster node. You can use existing EIPs. If no EIP is available, click **Auto create** to create one. When EIPs are automatically created, you are advised to set the bandwidth to a large value to avoid affecting the pressure test. Additionally, an EIP with specified configurations is automatically assigned to each node. If the number of EIPs is less than the number of nodes, the EIPs are randomly bound to the nodes.
- In the **(Optional) Advanced Settings** area, set **Kubernetes Node Name** to **Node private IP**. If you select **Cloud server name**, the node cannot be managed.

Click **Next: Confirm**. After confirming that the node configuration is correct, select **I have read and understood the preceding instructions** and click **Submit**. After the node is created, return to the CodeArts PerfTest console.

Step 5 Create a resource group.

Choose **Resource Groups** in the navigation pane, and click **Create Resource Group**.

Set the basic information by referring to **Table 3-2**.

Table 3-2 Creating a private resource group

Parameter	Description
Resource Group Name	Name of the private resource group to be created.
Debugging Cluster	Select a CCE cluster from the drop-down list.
Debug Node	Node used for debugging. It cannot be changed after the resource group is created. At least two debugging nodes need to be created.
Execution Node	Target machine that a pressure test will be performed on and can provide performance data during testing.

Click **Create**.

----End

3.3.2 Managing a Private Resource Group

After a private resource group is created, you can perform the following operations on it.

Custom JMeter Installation Package

To switch the JMeter version, customize the JMeter installation package to meet service requirements.

Step 1 Log in to the CodeArts PerfTest console and choose **Resource Groups** in the navigation pane.

Step 2 Download **JMeter installation package** of the required version.

Step 3 In the resource group list, click **Customization JMeter Installation package**, import the installation package, and wait until a message is displayed indicating that the import is successful. The JMeter installation package must be in .tgz, .tar.gz, or .zip format and smaller than 100 MB. The version must be 5.2.1 or later. Ensure that the installation package can be executable locally.

Step 4 (Optional) Delete the JMeter installation package.

If a message is displayed indicating that the upload fails, or you need to upload the JMeter installation package again, click  to delete the current package.

----End

Scaling a Node Cluster

Step 1 Log in to the CodeArts PerfTest console and choose **Resource Groups** in the navigation pane.

Step 2 In the resource group card, click **Resource Adjust**.

Step 3 Scale a node cluster.

- Scale out the resource group: Click **Scale**, select existing nodes and new nodes in the **Select Execution Node** dialog box, and click **OK**.
- Scale in the resource group: Click **Scale** under the resource group, select the execution nodes you want to keep, and click **OK**.

----End

Viewing Information About a Private Resource Group

Step 1 Log in to the CodeArts PerfTest console and choose **Resource Groups** in the navigation pane.

Step 2 In a resource group card, you can view the information listed in the following table.

Table 3-3 Private resource group information

Name	Description
Private Resource Group Name	Name entered when you were creating a private resource group.

Name	Description
Private Resource Group Status	Running Abnormal Upgradable Deploying
Actuators	In-use/Total Actuators Concurrency supported by a single executor is 5,000 (HTTP/HTTPS), 5,000 (WebSocket), 5,000 (MQTT), 1,000 (JMeter), or 1,000 (HLS/RTMP/HTTP-FLV).
Current Concurrency	Sum of concurrent users of all node clusters in the current private resource group.
Debug Node	Debugging nodes selected when you were creating a private resource group.
Execution Node	Execution nodes of all node clusters in a resource group. Click **More** to view all execution nodes of the resource group.
Created At	Time when the private resource group was created.
Modified At	Last time when the private resource group was modified.

----End

Deleting a Private Resource Group

Step 1 Log in to the CodeArts PerfTest console and choose **Resource Groups** in the navigation pane.

Step 2 In a resource group card, click **Delete**.

NOTE

When you delete a resource group, nodes will not be used in pressure tests again instead of being deleted. To delete a node, go to the CCE console.

----End

3.4 PerfTest Project Management

3.4.1 Creating a Test Project

CodeArts PerfTest helps you manage test projects. Transaction modes, test cases, test tasks, real-time reports, and offline reports are shared in a test project. You can create different test projects for different test programs.

You can customize or use a template to create a test project. CodeArts PerfTest defines templates for the following scenarios:

- **All-in-one network office system scene**: You can quickly construct a pressure model by simulating system access requests to detect service performance bottlenecks under different pressure models and prevent service breakdown.
- **Stream scene**: Supports streaming push and pull with common streaming protocols. Construct corresponding pressure test scenarios and simulate typical audio and video scenarios to perform performance pressure tests to identify risks in advance.
- **Seckill scene**: Simulate a large number of users performing operations on offerings at a specified time point to check the reliability of e-mall services when the access pressure increases exponentially.
- **E-commerce scene**: Simulate a large number of merchants logging in to the e-commerce system and perform operations such as viewing and maintaining offering information. Verify that related operations are not interrupted and services can run properly.
- **Quick Create**: Quickly create customized pressure test scenarios and orchestrate test cases on the test case page, greatly improving test case design efficiency.

Manually Creating a Test Project

Step 1 Log in to the CodeArts PerfTest console, choose **PerfTest Projects** in the left navigation pane, and click **Create Test Project**.

Step 2 Set the basic information by referring to **Table 3-4**.

Table 3-4 Creating a test project

Parameter	Description
Project Name	Name of a new test project. The project name can contain a maximum of 128 bytes, including letters, digits, hyphens (-), underscores (_), and periods (.).
Description	Description of a new test project.

Step 3 When the configuration is complete, click **OK**.

After creating a test project, you can add a test case for the test project. For details, see **PerfTest Case Management**.

----End

Creating a Test Project Using a Template

Step 1 Log in to the CodeArts PerfTest console and choose **Dashboard** in the left navigation pane.

Step 2 Select a template to automatically create test tasks. CodeArts PerfTest provides the following templates:

- **All-in-one Scene**
- **Stream Scene**
- **Seckill Scene**
- **E-commerce Scene**
- **Quick Create**

You can modify a test case as required. For details, see **PerfTest Case Management**.

----End

3.4.2 Managing Test Projects

After a test project is created, you can modify, delete, import or export it.

Modifying and Deleting a Test Project

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 In the PerfTest project list, choose  > **Modify Project**. Enter the project name or description and click **OK**.

Step 3 In the PerfTest project list, choose  > **Delete Project**. Delete the project as prompted.

----End

Exporting a Test Project

You do not need to write a test project from scratch. Instead, find a project similar to the service model, export the test project, slightly modify it, and import the test project.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Select one or more projects to be exported from the PerfTest project list, and click **Export Project**. Then save the exported test project file **ProjectList(***Serial number***).json** to the local host.

- If SLA rules have been configured for a test project, the SLA rules will not be exported when you export the project.
- If a file is uploaded for a global variable, the corresponding file will not be exported when you export the test project.
- Projects that use the MQTT protocol cannot be exported.

Step 3 Modify parameters and the project name (the name must be unique) in the file as required, and save the file. Do not change the format of the **ProjectList(***Serial number***).json** file.

----End

Importing a Test Project

Test projects of PerfTest can be imported to CodeArts PerfTest.

Step 1 Log in to the CodeArts PerfTest console, choose **PerfTest Projects** in the left navigation pane, and click **Import Project**.

Step 2 Import a PerfTest project. Currently, projects that use the MQTT protocol cannot be imported.

1. Set **Project Type** to **PerfTest project import**.
2. Click **Select File**, select a project file in JSON format, and click **Import**.

   The name of the imported project must be unique in CodeArts PerfTest. Otherwise, the project cannot be imported.

Step 3 After the project is imported, click **Close**. You can modify the test project by referring to **Modifying and Deleting a Test Project** and **Managing Transaction Request Information**.

----End

Viewing a Test Project

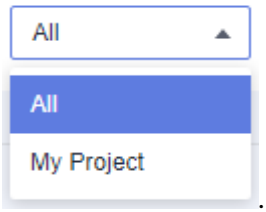
You can view the existing projects.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Enter a test project name in the search box 

in the upper right corner to search test projects.

You can also select a range to view test projects from the drop-down list



- **All**: Displays all projects within the current user's permission scope, including projects created by subaccounts.
- **My project**: Displays only projects created by the current user.

----End

3.5 PerfTest Case Management

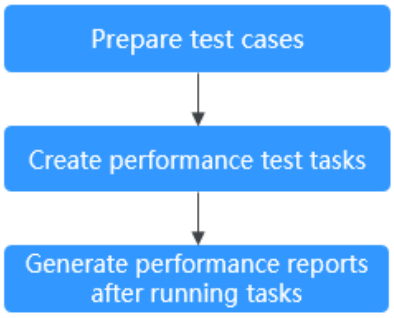
3.5.1 Test Case Description

3.5.1.1 Introduction to Test Cases

A PerfTest project consists of test cases, test tasks, and performance reports. Transactions are enhanced capabilities.

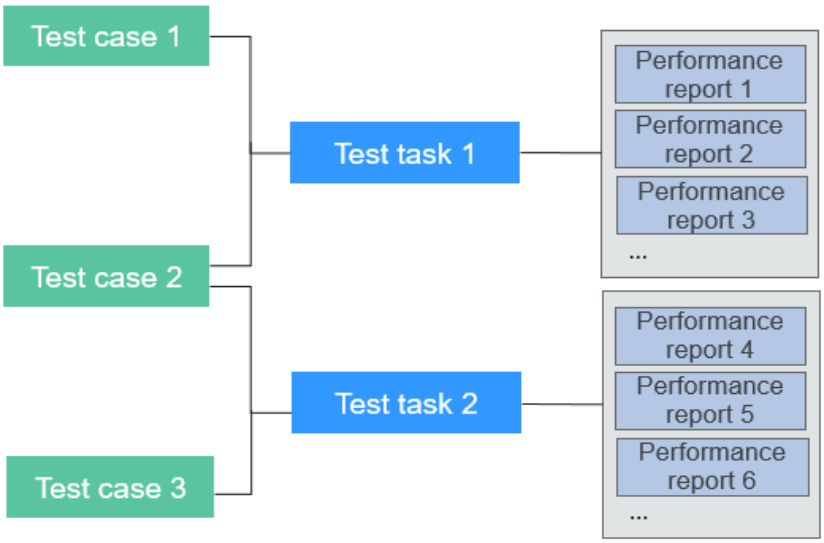
Test cases, test tasks, and performance reports are the three main steps of performance testing.

Figure 3-2 Performance testing process



A test case is a test model established based on a performance pressure test scenario. A test task is a performance test based on defined test models. A performance report is the execution result of a test task and displays the performance indicators of the tested system in high-concurrency scenarios.

Figure 3-3 Relationships between test cases, test tasks, and performance reports



3.5.2 Directory Management of Test Cases

A test case directory is a display structure for organizing and managing test cases in a test project.

Prerequisites

A **PerfTest project** has been created.

Creating a Test Case Directory

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

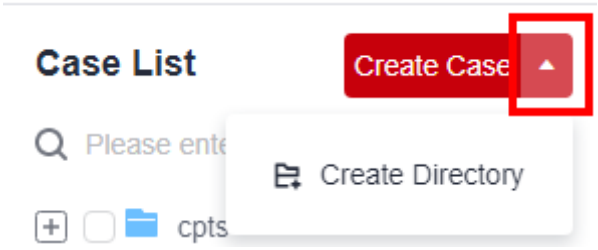
Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, create a directory.

- Create a case directory in the root directory.

  Click the arrow on the right of **Create Case**. Click **Create Directory** from the drop-down list and enter the directory name.

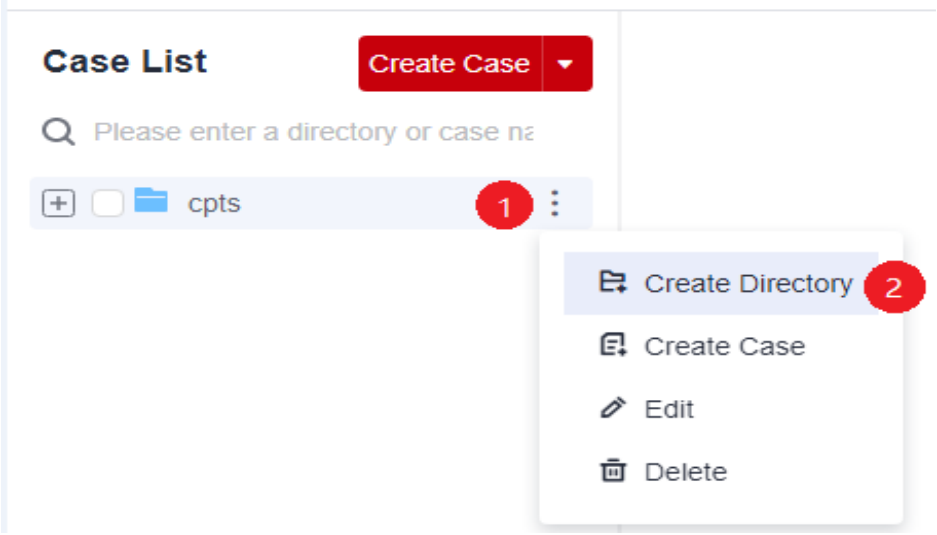
  Figure 3-4 Creating a case directory in the root directory

- Create a subdirectory in an existing directory.

  Click next to an existing directory, choose **Create Directory** from the drop-down list, and enter the directory name.

Figure 3-5 Creating a subdirectory in an existing directory



----End

Modifying a Test Case Directory

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, view the created case directory structure on the left of the page.

Step 4 Click next to the directory to be modified, and choose **Edit** from the drop-down list.

Step 5 Enter the new directory name in the text box and click another position on the page. The directory name is automatically saved.

----End

Deleting a Test Case Directory

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, view the created case directory structure on the left of the page.

Step 4 Click next to the directory to be deleted, choose **Delete** from the drop-down list, and delete the directory as prompted.

----End

3.5.3 Creating a Test Case

A test case is a test model established based on a performance pressure test scenario.

Prerequisites

- A **PerfTest project** has been created.
- A **test case directory** has been created.

Procedure

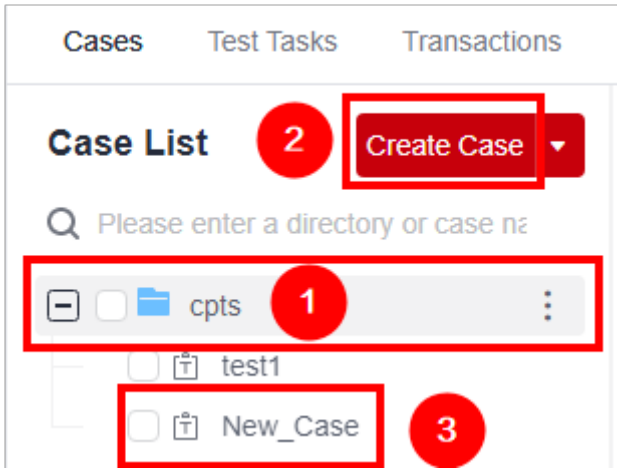
Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, view the created case directory structure on the left of the page. You can create a case in either of the following ways:

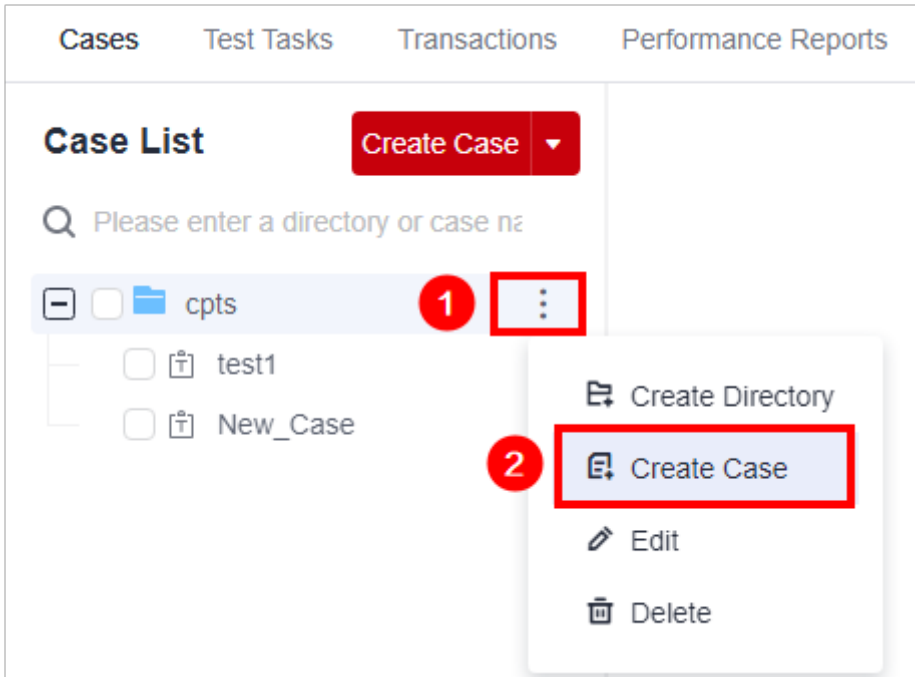
- To create a case in the current directory, click **Create Case**.

  Figure 3-6 Creating a case in the current directory

- To create a case in a specified directory, click next to the directory, and choose **Create Case** from the drop-down list.

Figure 3-7 Creating a case in the specified directory



Step 4 Click **Save**.

----End

3.5.4 Configuring a Test Case

3.5.4.1 Configuring a Case Script

After a test case is created, you can modify it as required.

Creating a Case Script

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 Click **Case Script** and select **Pre-Steps** or **Steps**.

Pre-Steps are executed only once in each execution thread. This configuration is optional.

Steps are executor of a test case, which needs to be configured.

Step 5 Add requests, transactions, data sets, cycle controllers, condition judgments, and rendezvous points as required.

You can add both requests and transactions, or add either of them. In practice, data sets, cycle controllers, condition judgments, and rendezvous points are used together with requests.

- For details about how to add requests, see **Adding Request Information (Packet)**, **Adding Request Information (Think Time)**, **Adding Request Information (Response Extraction)**, and **Adding Request Information (Checkpoint)**.
- Add a transaction. A transaction mode already exists on the **Transactions** tab page. Click **Add Transaction**, select a transaction name, and click **OK**. For details about how to create a transaction, see **Adding a Transaction**.
- For details about how to add data sets, cycle controllers, condition judgments, or rendezvous points, see **Adding a Data Instruction, Cycle Controller, Condition Judgment, or Rendezvous Point**.

Step 6 When the configuration is complete, click **Save**.

----End

Skipping a Request Upon Failure

When a request fails during case execution, the case does not report an error to terminate the execution. Instead, the case continues to execute subsequent requests. The failure of a request does not affect the execution of subsequent key steps.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

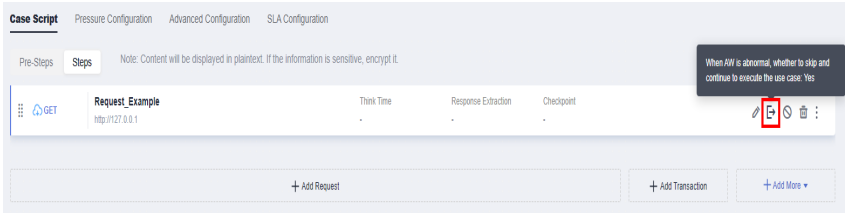
Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

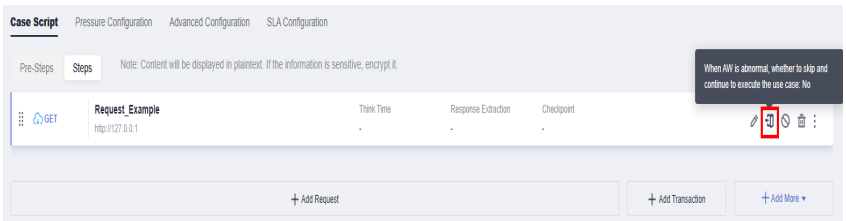
Step 4 On the **Case Script** tab page, find the desired request (AW).

Step 5 Click the second button ( ) on the right of the request to configure whether to skip the AW request upon failure. The default value is **No**.

- Yes: Skip a failed request. The case execution is not interrupted.



- No: By default, a failed request is not skipped. The request failure does not interrupt case execution.



----End

Samples

If request **aw-1** is configured to be skipped upon failure, the system will skip the request failure and continue to execute the subsequent request **aw-2**.

Figure 3-8 AW that is configured to be skipped upon failure

Retrying a Request Upon Failure

When a request fails during case execution, the system retries the request based on the configuration, including the retry times and retry interval. If a request fails to be executed due to network or performance errors, configure retries for fault tolerance.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 On the **Case Script** tab page, find the desired request (AW).

Step 5 On the **Checkpoint** tab, enable **Enable Result Check** to configure retry upon failure.

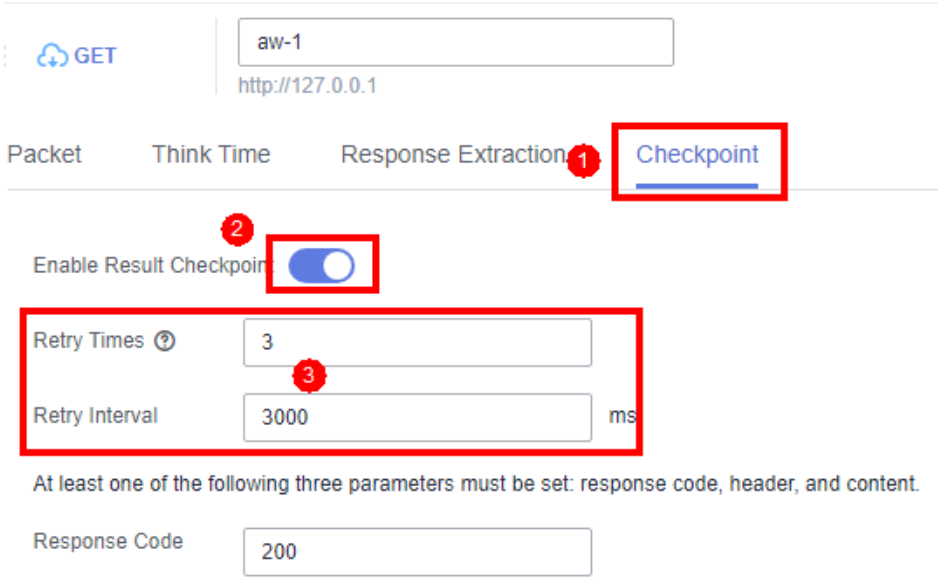
----End

Samples

aw-1 is configured to be retried upon failure.

- If the **aw-1** request is successfully executed, the **aw-2** request will be executed without retrying **aw-1**.
- If the **aw-1** request fails, it will be retried three times at an interval of 3000 ms based on the checkpoint.
  - If the **aw-1** request is successful during the three retries, the **aw-2** request will be executed.
  - If all the three retries fail, the **aw-1** request fails, and the **aw-2** request will be interrupted.

Figure 3-9 Checkpoint enabled for the aw-1 request



Disabling/Enabling a Case Request

If you want to retain a request of a case but do not want to use it, you can disable it. If you want to resume the use of the request, you can enable it again.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 Disable a case request.

On the **Case Script** tab, click next to the desired request to disable it. Other requests are not affected.

Step 5 Enable a case request.

On the **Case Script** tab, click next to the desired request to enable it. Other requests are not affected.

----End

Copying/Pasting a Case Request

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 On the **Case Script** tab, click > **Copy** next to the desired request. **The copy is successful!** is displayed in the upper right corner of the page.

Step 5 You can select a request in any project under **PerfTest Projects**, click > **paste to front** or **paste to behind** next to the request.

----End

3.5.4.2 Adding Request Information (Packet)

Packets are data blocks transmitted between HTTP/HTTPS/TCP/UDP/HLS/RTMP/WebSocket/HTTP-FLV/MQTT-based applications. These data blocks start with text metadata that describes the packet content and meaning. The text metadata is followed by optional data. Packets are transmitted among clients, servers, and agents.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 On the **Case Script** tab page, select a case, and click **Add Request**.

Step 5 On the **Packet** tab page, select a protocol type based on the service protocol type. Currently, HTTP, HTTPS, TCP, UDP, HLS, RTMP, WebSocket, MQTT, and HTTP-FLV are supported. Set the basic information according to the following packet parameter tables.

Table 3-5 HTTP and HTTPS packet parameters

Parameter	Description
Request mode	**GET**, **POST**, **PATCH**, **PUT**, or **DELETE**.
Request URL	URL for sending a request, for example, **http://***domain name***/***path*, or **http://***domain name***/path?key1=value1&key2=value2**. Variables can be inserted. For details, see **Inserting a Variable**.
Response Timeout (ms)	Timeout period for waiting for a response from the server when a request is sent. The timeout ranges from 20 ms to 60s. If this parameter is not set, the default response timeout interval (5,000 ms) is used.
Carried cookie	At least one request exists in the case. Set this parameter when adding new requests. • **Obtain automatically**: Use the cookie set in the response. • **Obtain manually**: Use the cookie in the header of the current request.

Parameter	Description
Request Parameter	Set the parameters in the request URL. Click **Add Request Parameter** and set **Name** and **Value**. If you do not need to add a request parameter, click **Delete** to delete it.
RequestHeader	Add headers and contents based on the headers to be verified or used by a pressure test server. Headers are not mandatory. Only user-defined headers are transparently transmitted to the pressure test server. For details about **Header**, see **Header Description**. Click **Add Header** and set **Header** and **Value**. If you do not need to add header information, click **Delete** to delete the packet header. You can click **Bulk Edit** to edit multiple headers at a time in the text box. You can click **Key-Value Edit** to return to the key-value pair editing mode. If the request mode is **POST** or **PUT** and **Header** is **Content-Type**, variables can be inserted, and **Value** can be one of the three following types: • **Customize**: Enter a value in the text box. • **application/x-www-form-urlencoded**: The request body is a key value pair that can be added. The value is text. • **multipart/form-data**: The request body is a key-value pair that can be added. The value can be text or a file. If the value is a file, the body can be imported.
Body	The body of an entity contains a data block consisting of random data. Not all packets contain the body of an entity. When the request mode is **GET**, packet content cannot be input. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values. 1. Enter **$** in the request content input box. 2. In the **Insert Variable** dialog box, set parameters. For details, see **Inserting a Variable**. 3. Click **Select**.

Table 3-6 TCP packet parameters

Parameter	Description
IP	IP address of the tested server to which requests are sent.
Port Number	Port number of the tested server to which requests are sent.
Connection Timeout	Timeout duration for the server's response after a connection is initiated. The connection timeout value ranges from 20 ms to 60s.
Response Timeout	Timeout duration for waiting for the server's response after a connection is established. The response timeout value ranges from 20 ms to 60s.
Connection Settings	• **Repeated use**: When a request response is complete, the connection remains and is used to send and receive the next request response. • **Close**: When a request response is complete, the connection is closed and re-established next time.
Return Settings	Used to judge whether a request response has been received. It is recommended that a unique end-of-text character be set. If multiple such characters are present in a response, the response is considered complete when the first character is received. As a result, the received response data may be incomplete. • **Length of returned data**: length of the returned data, in bytes. When a response of this length is received, data receiving is complete. • **End-of-text character**: ending mark of the returned data. When an end-of-text character is received, data receiving is complete.
Content Format	Select **String** or **Hexadecimal code stream data** based on the service request content of the tested server. A hexadecimal code stream can contain only numbers 0–9 and letters a–f. The total number of characters must be an even number. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values. 1. Enter **$** in the request content input box. 2. In the **Insert Variable** dialog box, set parameters. For details, see **Inserting a Variable**. 3. Click **Select**.

Table 3-7 UDP packet parameters

Parameter	Description
IP	IP address of the tested server to which requests are sent.
Port Number	Port number of the tested server to which requests are sent.
Return Settings	Used to judge whether a request response has been received. It is recommended that a unique end-of-text character be set. If multiple such characters are present in a response, the response is considered complete when the first character is received. As a result, the received response data may be incomplete. • **Length of returned data**: length of the returned data, in bytes. When a response of this length is received, data receiving is complete. • **End-of-text character**: ending mark of the returned data. When an end-of-text character is received, data receiving is complete.
Content Format	Select **String** or **Hexadecimal code stream data** based on the service request content of the tested server. A hexadecimal code stream can contain only numbers 0–9 and letters a–f. The total number of characters must be an even number. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values. 1. Enter **$** in the request content input box. 2. In the **Insert Variable** dialog box, set parameters. For details, see **Inserting a Variable**. 3. Click **Select**.

Table 3-8 HLS packet parameters

Parameter	Description
Streaming media address	Address of the video source, for example, **http://** *domain name*/*path*.
Play Duration (s)	Duration of simulated playback. During case debugging, the playback duration is set to 3 seconds.
Retry Delay (ms)	Interval for obtaining new data in live streaming when the video playback source does not have new data.

Parameter	Description
Retry Times	Maximum number of attempts to obtain new data in live streaming when the video playback source does not have new data. If the number of attempts exceeds the maximum, the system determines that the attempt fails.

Table 3-9 RTMP packet parameters

Parameter	Description
Request Mode	• **Stream pushing**: a process of transmitting onsite video signals to the network. • **Stream pulling**: a process of pulling streaming media video files from a server using a specified address.
When **Request Mode** is **Stream pushing**, set the following parameters.	
Streaming server address	IP address of the tested streaming media server.
Port Number	Listening port number of the tested streaming media server. The default value is **1935** for RTMP.
App name	AppName in the stream URL, that is, the path for storing live streaming media files, for example, **live**.
Streaming name	StreamName in the stream URL, which uniquely identifies a live stream, for example, **livestream**.
Push duration (s)	Duration for simulating stream pushing, in seconds.
Video source address	OBS address for storing video source files for simulating live stream pushing. Currently, only .flv files are supported. Non-standard H.265-encoded .flv files are not supported.
When **Request Mode** is **Stream pulling**, set the following parameters.	
Streaming server address	IP address of the tested streaming media server.
Port Number	Listening port number of the tested streaming media server. The default value is **1935** for RTMP.
App name	AppName in the stream URL, that is, the path for storing live streaming media files, for example, **live**.
Streaming name	StreamName in the stream URL, which uniquely identifies a live stream, for example, **livestream**.

Parameter	Description
Play Duration (s)	Simulated duration of live stream watching, in seconds.

Table 3-10 WebSocket packet parameters

Parameter	Description
Request mode	**Connect**: Establish a WebSocket connection with the pressure test website. **DisConnect**: Close the connection to the pressure test website. **Pong**: Send **pong** to the pressure test website. **Ping&Pong**: Send **pong** to the pressure test website and expect to return **ping**. **Request**: Send a request to the pressure test website and receive a response. **ReadOnly**: Only receive information sent by the pressure test website. **WriteOnly**: Only send information to the pressure test website.
Request connection address	URL of the WebSocket connection, for example, **ws://***domain name*/*path*. Encrypted requests are supported, such as **wss://***domain name*/*path*.
Connection Timeout (ms)	This parameter is available when the request mode is **Connect**. Timeout duration for the server's response after a connection is initiated. The connection timeout value ranges from 20 ms to 60s.
Status Code	This parameter is available when the request mode is **DisConnect**. Status code for closing a connection. The default value is **1,000**. The value range is 0–4,999.
Request Type	This parameter is available when the request mode is **Request** or **WriteOnly**. Select **Text** or **Binary** based on the service request content of the tested server. **Binary** indicates hexadecimal.

Parameter	Description
Body	This parameter is available when the request mode is **Request** or **WriteOnly**. Enter the packet content of request information based on the specified request type. When the request content is text, you can enter variables. For details, see **Inserting a Variable**.
Response Type	This parameter is available when the request mode is **Request** or **ReadOnly**. Select **Text** or **Binary** based on the service response content of the tested server. **Binary** indicates hexadecimal.
Response Timeout (ms)	Timeout duration for waiting for the server's response after a connection is established. The response timeout value ranges from 20 ms to 60s.
Request Header	Add headers and contents based on the headers to be verified or used by a pressure test server. Headers are not mandatory. Only user-defined headers are transparently transmitted to the pressure test server. For details about **Header**, see **Header Description**. Click **Add Header** and set **Header** and **Value**. If you do not need to add header information, click **Delete** to delete the packet header. You can click **Bulk Edit** to edit multiple headers at a time in the text box. You can click **Key-Value Edit** to return to the key-value pair editing mode.

Table 3-11 HTTP-FLV packet parameters

Parameter	Description
Streaming media address	Address of the video source, for example, **http://** *{Video source IP address}*:**8080/live/stream.flv**.
Play Duration (s)	Duration of simulated playback. During case debugging, the playback duration is set to 3 seconds.

Table 3-12 MQTT packet parameters

Parameter	Description
Request mode	**Connect**: Establish an MQTT connection with the pressure test website. **Publish**: Publish a message. **Subscribe**: Subscribe to sent messages. **DisConnect**: Close the MQTT connection with the pressure test website.
When the request mode is **Connect**, set the following parameters.	
MQTT Version	MQTT version. Currently, 3.1 and 3.1.1 are supported.
Transport Address	MQTT server address, which supports the TCP, SSL, WebSocket, and WSS protocols. Domain names cannot be bound to MQTT transmission addresses.
Timeout interval	Timeout interval for the client to perform operations such as establishing a connection and sending a message.
Client ID	MQTT client ID. You can set whether to add a random suffix to the client ID by enabling or disabling **Add a random suffix**.
Username	Username set for the MQTT client. This parameter is sensitive. Set it using global variables in sensitive parameter mode.
Password	Password set for the MQTT client. This parameter is sensitive. Set it using global variables in sensitive parameter mode.
Connection keepalive	Active heartbeat interval, in seconds.
Clear Sessions	• If this option is toggled on, the reserved session information of the last connection will be cleared when a connection is established between the client and the server. • If this option is toggled off, the reserved session information of the last connection will not be cleared when a connection is established between the client and the server.

Parameter	Description
Add Will	• You can toggle on this option to add a will. In this way, when the client is disconnected abnormally, the MQTT agent automatically publishes the will message to other subscribers. Set the following parameters when adding a will: – **Theme**: topic of the will. – **Type**: **Text** or **Hex**. You can also set whether to add a timestamp. – **Content**: content of the will. – **QoS Level**: quality of service (QoS) level, which is used to describe the reliability of message transmission. **Send at most once**: applicable to real-time data transmission. **Send at least once**: applicable to applications that require reliability but allow a slight delay. **Only once**: applicable to applications that require high reliability and allow a long delay. – **Keep your will**: If this option is toggled on, the subscribing client receives the set will message immediately after subscription. If this option is toggled off, the subscribing client receives the will message only after the client that publishes the will message is disconnected unexpectedly. • If you toggle off **Add Will**, no will message will be sent.
If the request mode is **Publish**, set the following parameters:	
Theme	Topic of the message to be published.
QoS Level	QoS level, which is used to describe the reliability of message transmission. **Send at most once**: applicable to real-time data transmission. **Send at least once**: applicable to applications that require reliability but allow a slight delay. **Only once**: applicable to applications that require high reliability and allow a long delay.
Request Type	**Text** or **Hex**. You can also set whether to add a timestamp.
Request Content	Detailed content of the request.
Timeout interval	Timeout interval for the client to send a message.
Reserved Message	• If this option is toggled on, messages published when the client is disconnected are stored on the server. • If this option is toggled off, messages published when the client is disconnected are cleared.

Parameter	Description
When the request mode is **Subscribe**, set the following parameters.	
Subscribe to Topic	Topic of the subscribed message, which must match the topic of the published message.
QoS Level	QoS level, which is used to describe the reliability of message transmission. **Send at most once**: applicable to real-time data transmission. **Send at least once**: applicable to applications that require reliability but allow a slight delay. **Only once**: applicable to applications that require high reliability and allow a long delay.
Response Type	**Text** or **Hex**. You can also set whether to add a timestamp.
Timeout interval	Timeout interval of the subscription topic.
End Condition	When this condition is met, the subscription ends. • **Maximum triggering time**: The subscription ends when the time reaches the upper limit. • **Maximum Number of Received Messages Triggered**: The subscription ends when the number of received messages reaches the upper limit.
If the request mode is **Disconnect**, you do not need to set parameters.	

Step 6 When the configuration is complete, click **Save**.

----End

Filling in a Packet

Packets refer to all click operations performed by users on the website interface. The process of sending a packet: A click operation is edited to a code stream complying with protocol specifications and carrying a user's request before the code stream is sent to a third party, leading to a correct or failed response.

Press **F12** or use a packet-capturing tool (such as Wireshark) to check how a packet is requested and fill in the packet to be tested according to the actual service.

3.5.4.3 Adding Request Information (Think Time)

To better simulate user behavior, insert a waiting time between different operations. For example, when a user receives data from the server, the user may wait several seconds before viewing data and providing responses. This period of time is called think time.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 On the **Case Script** tab page, select a case, and click **Add Request**.

Step 5 On the **Think Time** tab page, set the basic information by referring to **Table 3-13**.

Table 3-13 Think time parameters

Parameter	Description
Enable Think Time	After think time is enabled, you may wait for several seconds before viewing the data received from the server, and then respond.
Duration (ms)	Think time refers to the waiting time between two consecutive operations of a user, such as the interval between login and search. For example, the response time for each case execution is 0.5s. • To execute two case requests per second, do not add a case request whose type is **think time**. • To execute only one case request per second, set **Duration** to 1s. If the think time is set to 1s and the response time is longer than 1s, the think time will not take effect, and pressure test requests are sent according to the response time.

Step 6 When the configuration is complete, click **Save**.

----End

3.5.4.4 Adding Request Information (Response Extraction)

If multiple packets exist in the same case, use regular expressions or JSON to extract the output of the previous packet for the input of the next packet.

For example, in an e-commerce flash sale scenario, after you search for a product and purchase it, you can use the response extraction function to extract the product ID obtained from packet search and use the ID as the input parameter of the next purchase packet.

Notes and Constraints

- Cross-case extraction is not supported. That is, the output extracted from case A's packet cannot be used for the input of case B's packet.
- The variables from a response extraction are local variables.
- After an array is extracted from a response variable, the array can only be used as the input of the array variable in the next packet. Each field in the array cannot be used separately.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 On the **Case Script** tab page, select a case, and click **Add Request**.

Step 5 On the **Response Extraction** tab page, set the basic information by referring to **Table 3-14**.

Table 3-14 Response extraction parameters

Parameter	Description
Enable Response Extraction	After response extraction is enabled, if multiple packets exist in the same case, you can use regular expressions or JSON to extract the output of the previous packet for the input of the next packet.
Variable Name	Must be unique. The value of the response extraction is assigned to this variable.
Expected Value	Expected variable value of response extraction. If you enable this option, the extracted value is compared with the expected value. If they are different from each other, the URL verification will fail.
Extraction Range	Range of the content to be extracted. The packet content, header, and URL can be extracted using regular expressions.
Regular Expression	A regular expression specifies a type of logical expressions performed on strings. In a regular expression, you use certain predefined strings and a combination of these strings to form a rule string that is used to specify a specific filtering logic. Parentheses **()** indicate extraction and are used to enclose content to be extracted. The content in a pair of parentheses forms a sub-expression. A complete regular expression consists of two types of characters: special characters (also called meta characters) and literal or common text characters (such as letters, digits, and underscores). For details about meta characters, see **Regular Expression Metacharacters**.
Sequence Number of Matching Item	This parameter indicates the sequence number of the matched content extracted through a regular expression. This parameter cannot be set to **0**. Value range: positive integers

Parameter	Description
Expression Value	This parameter indicates the sequence number of the parsed sub-expression. • Value **0** indicates that the entire regular expression is matched. • Value **1** indicates that the first sub-expression of the regular expression is matched, that is, the content extracted by the first **()**. Value range: natural numbers After extracting content through **Regular Expression** and **Sequence Number of Matching Item**, use **Expression Value** to obtain the final content.
Key Name	Enter the key name to be obtained. This parameter is valid only when **Extraction Range** is set to **Parameter values in .json files**. For example, to extract **v42** from {"key":{"key1":"v1","key2":{"key3":"v3"},"key4":[{"key41":"v41","key42":"v42"},{"key41":"v43","key42":"v44"}]}}, enter **key.key4[0].key42**.
Default Value	Value extracted when a regular expression match or JSON extraction fails.
Condition Expression	This parameter is used together with the **Key Name** to be obtained. For example, {"key":{"key1":"v1","key2":{"key3":"v3"},"key4":[{"key41":"v41","key42":"v42"},{"key41":"v43","key42":"v44"}]}}. If you want to describe when key42 is equal to v42, extract the target value **v41**, you can enter **key.key4[].key42 = v42** for **Condition Expression**, and enter **key.key4[].key41** for **Key Name**.

Table 3-15 Common regular expressions

Regular Expression	Description	Example
(\d+)	Matches non-negative integers.	String: bTivm2wu9jih1LBKR4osZGrjjl
		Match results: 2 9 1 4
([A-Za-z]+)	Matches a string of letters.	String: bTivm2wu9jih1LBKR4osZGrjjl

Regular Expression	Description	Example
		Match results: bTivm wu jih LBKR osZGrjjl
([A-Za-z0-9]+)	Matches a string of digits and letters.	String: bTivm2wu9jih1LBKR4osZGrjjl:asdasd22
		Match results: bTivm2wu9jih1LBKR4osZGrjjl asdasd22
(\w+)	Matches a string of digits, letters, or underscores (_).	String: bTivm2wu9jih1LBKR4osZGrjjl:asdasd22
		Match results: bTivm2wu9jih1LBKR4osZGrjjl asdasd22
([\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+)	Matches email addresses.	String: bTivm2wu9jih1LBKR4osZGrjjl:abc@abc.com
		Match results: abc@abc.com

Step 6 (Optional) A request can contain one or more variables. If you need to add multiple variables for a request, click **Add Variable** and set related parameters.

Step 7 When the configuration is complete, click **Save**.

----End

Examples of Regular Expression Response Extraction

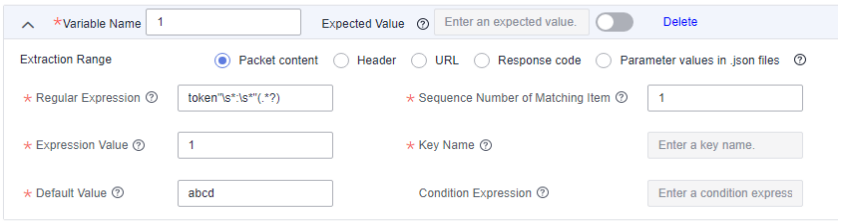
Example 1

The response content of the previous packet is as follows:

```
"baseInfo" : {
"mobilephone":"xxxxxxxxxxx",
"Telephone":"xxxxxxxxxxx",
"unitGuid":"xx",
"unitMame":null,
"address":"xxx",
"gender" : 0,
"imageUr1" :nul1
},
"UserNotices":null
},
"msgId'" : "64xxxxxxxxxxxxxxxxxxxxxxxf5",
"isUsed" :"1"
"token":"eyxxxxxxxxxxJ9.eyJzdW1101JYZMiLCJcUbdGUkIJezMT!!zz#z!
20TxxxxxxxxxxxxxxxxxxxS1611dYQISHTFKifVOsImd1aWQi0ilOODkiLCJleHAiOjEiMzIizNzY1MjZ9.myU5idiASM-1
1@EP7YQTfTsR_8zsq7?sbYJYoxfRsuf6OZhGL-
XWmjnvdaviGauhSdw16ImWOFEvbACSHMXGT1U0ijS5z6ezX@sZePruzFnvcIMgShF8xNPN6zVokQp-
uwbyS3W6NpZpDuwsjuiZ7DZTNpKoqCkGHwvPJrHBOrWFR_u6-FBbTiFiqdhQb95U-1gLiLvoZHY_rguzwyrZ-
leGRdCG_ZASreoWC-
uH)HnqltpgItrChWQToHQyxOABdMSbBSHhNctBBZHgQPMESqQQQTbBiPGvbQDprB7ZBFMUB_ShynS_evtyfE
```

```
ladGEddhOBn-fxxxxxxxxxx"
},
```

To extract the token value, use the regular expression **"token"\s*:\s*"(.*?)"**. The configuration of response extraction is shown as follows:

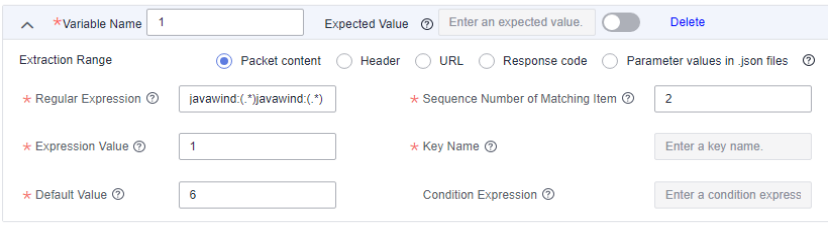


Example 2

The response content of the previous packet is as follows:

```
javawind:9javawind:12
javawind:16javawind:17
javawind:46javawind:22
```

To extract the value 16, see the following figure.

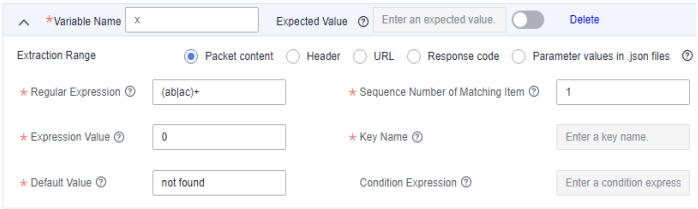


1. The following data is extracted using the regular expression **javawind:(.*)javawind:(.*)**.
   ```
   9 12
   16 17
   46 22
   ```
2. The following data is extracted using the second matching item.
   ```
   16 17
   ```
3. The following data is extracted using the first expression.
   ```
   16
   ```

Example 3

The packet content is **ababdacac**.

Response extraction settings:



Regular Expression: **(ab|ac)+** indicates that multiple **ab** or **ac** values are matched.

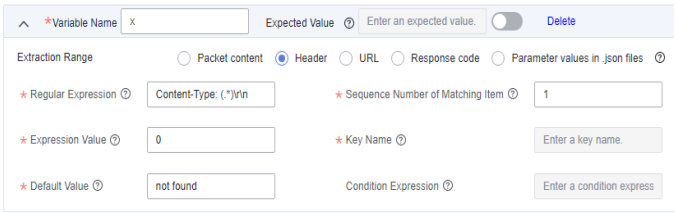
Sequence Number of Matching Item: **1**, indicating that the matching item is matched by **ab**. **2**, indicating that the matching item is matched by **ac**. **1** may be

used to obtain **abab** and a substring **ab**, and **2** may be used to obtain **acac** and a substring **ac**.

Expression Value: The value **0** indicates that the maximum matching string **abab** or **acac** is used. The value **1** indicates that the substring **ab** or **ac** is used.

Example 4

The extracted content is **Content-Type** in the HTTP response header.



Regular Expression: **Content-Type: (.*)\r\n**. Note that there is a space after the colon (:) and **\r\n** is at the end, which is generated based on HTTP specifications.

Sequence Number of Matching Item: **1**, because there is only one **(.*)**.

Expression Value: **1**, indicating that the expected value is obtained.

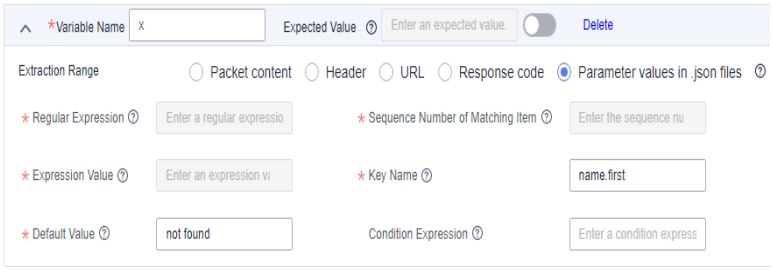
Examples of JSON Expression Response Extraction

The sample data is as follows:

```
{
   "name": {
           "first": "Tom",
           "last": "Anderson"
   },
   "age": 37,
   "children": ["Sara","Alex","Jack"],
   "fav.movie": "Deer Hunter",
   "friends": [{
           "first": "Dale",
           "last": "Murphy",
           "age": 44,
           "nets": ["ig","fb","tw"]
   },
   {
           "first": "Roger",
           "last": "Craig",
           "age": 68,
           "nets": ["fb","tw"]
   },
   {
           "first": "Jane",
           "last": "Murphy",
           "age": 47,
           "nets": ["ig","tw"]
   }]
}
```

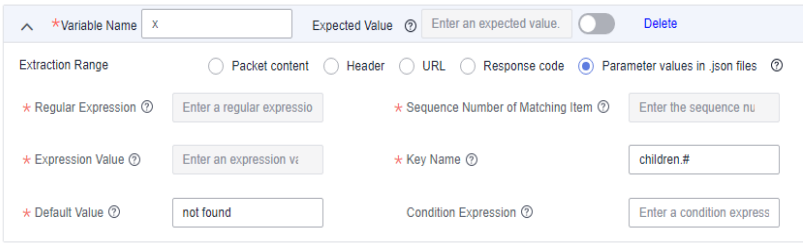
Example 1

To extract the **first name**, set **Key Name** to **name.first**.



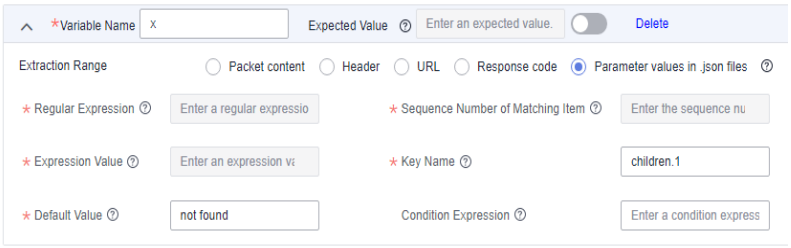
Example 2

To extract **the number of Tom's children**, set **Key Name** to **children.#**.



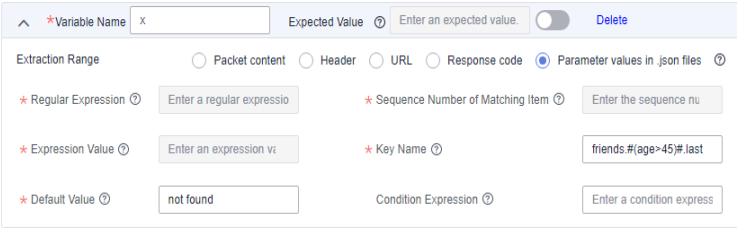
Example 3

To extract **the name of Tom's second child**, set **Key Name** to **children.1**. (Sequence number starts from 0.).



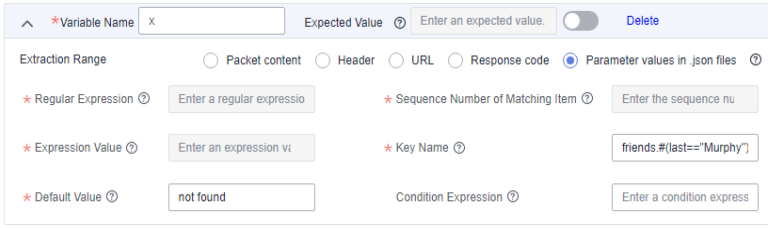
Example 4

To extract **the last names of Tom's friends older than 45**, set **Key Name** to **friends.#(age>45)#.last**.



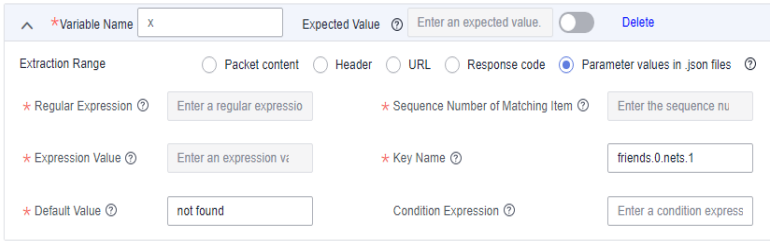
Example 5

To extract **the first name of Tom's friend whose last name is Murphy**, set **Key Name** to **friends.#(last=="Murphy")#.first**.



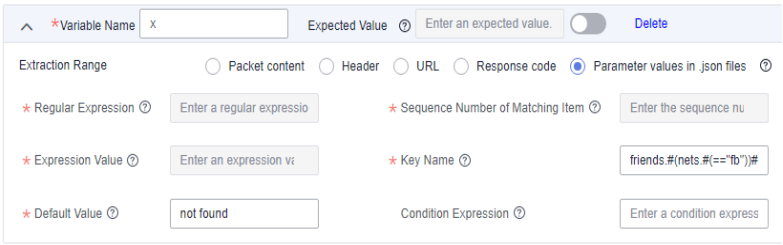
Example 6

To extract **the second nickname of Tom's first friend**, set **Key Name** to **friends.0.nets.1**. (Sequence number starts from 0.)



Example 7

To extract **the first name of Tom's friend whose nickname is fb**, set **Key Name** to **friends.#(nets.#(=="fb"))#.first**.



3.5.4.5 Adding Request Information (Checkpoint)

Checkpoint uses customized verification information to check whether the content returned by a server is correct. For different protocol types, the checkpoint supports different contents. The HTTP and HTTPS protocols support the verification of response code, header, and content. The TCP, UDP, MQTT, and WebSocket protocols support only content verification.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 On the **Case Script** tab page, select a case, and click **Add Request**.

Step 5 On the **Checkpoint** tab page, set the basic information by referring to **Table 3-16**.

Table 3-16 Checkpoint parameters

Parameter	Description
Enable Result Checkpoint	After this option is enabled, you can check whether the content returned by a server is correct through customized verification information.
Retry Times	Number of times to retry a failed request until it succeeds. The value ranges from 0 to 100. Statistics such as response time, RPS, and success rate of retried requests are not included in the report. The report counts only the last data.
Retry Interval	Interval for retrying a failed request. The unit is ms. The value ranges from 0 to 3,600,000, that is, the maximum interval is 1 hour.
Response Code	Set this parameter only when the protocol type is set to **HTTP** or **HTTPS** on the **Packet** tab page. This parameter is the HTTP or HTTPS response status codes, including **1XX**, **2XX**, **3XX**, **4XX**, and **5XX**, carried in a packet.
Header	Set this parameter only when the protocol type is set to **HTTP** or **HTTPS** on the **Packet** tab page. Headers of HTTP and HTTPS. 1. Click **Add Header**. 2. Enter a **Header Name**. For details, see **Header Description**. 3. Set check conditions. 4. Set contents.
Response Content	Body parts of HTTP, HTTPS, TCP, UDP, MQTT, and WebSocket, namely the load parts of the requests and responses. 1. Click **Add Content**. 2. Set check conditions.
Condition Rule	• **AND**: The verification is successful only when all the check conditions are met. • **OR**: The verification is successful when any check condition is met.

Step 6 When the configuration is complete, click **Save**.

----End

3.5.4.6 Adding a Data Instruction, Cycle Controller, Condition Judgment, or Rendezvous Point

Data Instruction

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 On the **Case Script** tab, choose **Add More** > **Data Set**.

Step 5 Click **Add Param**, and configure **Name**, **Logic**, and **Mode**.

- **Name**

  Name of the parameter to which the parameter logic result will be assigned as the parameter value.

- **Logic**

  a. To assign a constant value to a parameter, enter a character string, for example, **1.23** or **abc**.

  b. To assign a variable value to a parameter, enter **$** in the text box. In the displayed **Insert Variable** dialog box, select a variable type.

  If you set **Type** to **Custom**, set the following parameters:

  i. **Range**: variable range.

  ii. **Name**: name of a custom variable. Specifies the name of a variable when the variable is added.

  If you set **Type** to **System**, select a function. For details, see **Table 3-17**.

Table 3-17 Functions

Function	Description
ID card number	Randomly generates an ID number.
Mobile phone number	Randomly generates a phone number.
Random number in a range	Randomly generates an integer within your specified range.
Random character string	Randomly generates a string consisting of lowercase letters and numbers based on your specified string length (1 to 32 characters).
Time stamp	Generates a timestamp for the current time based on your specified unit. A 10-digit timestamp represents time in seconds whereas a 13-digit timestamp represents time in milliseconds.
UUID	Randomly generates a string of 32 characters.

If you set **Type** to **Four Fundamental rules**, select a function.

i. Select **+**, **-**, **x**, **/**, or **Scale Mode** from the drop-down list box, and click **Select**.

ii. Click  . In the displayed **Four Functions Editing** dialog box, set the following parameters:

- **Expression**: Enter operands for the arithmetic operation. An operand can be a number (such as 1.23), a global, local, or system variable. You can reselect the operation here.
- **Mode**: Select how the expression result will be rounded. For details, see **Table 3-18**.

Table 3-18 Rounding modes

Name	Description (Using Two Decimal Place Rounding as an Example)
Far-zero Rounding	Rounds the number in a way that the result heads away from zero. For example, 1.234 is rounded to 1.24, and –1.234 is rounded to –1.24.
Near-zero Rounding	Rounds the number in a way that the result heads towards zero. For example, 1.234 is rounded to 1.23, and –1.234 is rounded to –1.23.
Round Up	Rounds the number towards positive infinity. Positive numbers will be rounded up, whereas the unwanted digits of negative numbers will be simply discarded. For example, 1.234 is rounded to 1.24, and –1.234 is rounded to –1.23.
Round Down	Rounds the number towards negative infinity. The unwanted digits of positive numbers will be simply discarded whereas negative numbers will be rounded down. For example, 1.234 is rounded to 1.23, and –1.234 is rounded to –1.24.
Rounding	If the first digit to be discarded is smaller than 5, the digit and all the digits that follow are simply discarded. Otherwise, the last retained digit is increased by 1 in value. For example, 1.234 is rounded to 1.23 and –1.235 is rounded to –1.24.
Rounded to six	If the first digit to be discarded is smaller than 6, the digit and all the digits that follow are simply discarded. Otherwise, the last retained digit is increased by 1 in value. For example, 1.235 is rounded to 1.23 and –1.236 is rounded to –1.24.

Name	Description (Using Two Decimal Place Rounding as an Example)
Bankers	Rounds the number to its nearest even integer. For example. 1.2350000 is rounded to 1.24, 1.2250000 is rounded to 1.22, and 1.2250001 is rounded to 1.23.
No Rounding	The number is not rounded. If this mode is selected, no further action is required.

    - **Precision**: The maximum number of decimal places is 32.
  iii. When the configuration is complete, click **OK**. If you specify multiple logics for a parameter, all the logics are combined as a character string and the string is assigned to the parameter as its value.

- **Mode**
  - If you select **Repeated assignment** from the drop-down list box, a value is assigned each time the parameter value is needed in a thread.
  - If you select **Assigned once** from the drop-down list, the parameter value is assigned only once in a thread.

Step 6 When the configuration is complete, click **Save**.

----End

Cycle Controller

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired case in the case list on the left.

Step 4 On the **Case Script** tab, choose **Add More** > **Cycle Controller**.

Step 5 Set parameters.

Number of Cycles: Enter a value between 1 and 1,000,000.

Exit Conditions: If the request meets the condition, the cycle ends. Both expressions and comparisons support variables and character strings. Currently, conditions cannot be combined. If you need to use a set or multiple comparisons, set a regular expression.

Step 6 Click **Add More** to specify more details for the cycle. Up to 10 layers of steps can be nested.

Step 7 When the configuration is complete, click **Save**.

----End

Condition Judgment

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired case in the case list on the left.

Step 4 On the **Case Script** tab, choose **Add More** > **Condition Controller**.

Step 5 Set parameters.

Judgment Condition: If a request meets the condition, the request, transaction, or other logic controller defined by the condition controller is executed. Both expressions and comparisons support variables and character strings. Currently, conditions cannot be combined. If you need to use a set or multiple comparisons, set a regular expression.

Step 6 Click **Add More** to add requests, transactions, or other logic controllers that are executed after conditions are met. Up to 10 layers of steps can be nested.

Step 7 When the configuration is complete, click **Save**.

----End

Rendezvous Point

The purpose of a rendezvous point is to block concurrency until either X concurrent requests are blocked or the waiting time is reached, at which point all of them are released at a time. Therefore, a rendezvous point can create large instant loads at different points of the test case.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired case in the case list on the left.

Step 4 On the **Case Script** tab, choose **Add More** > **Rendezvous Point**.

Step 5 Set parameters.

Waiting Time (ms): waiting time at the rendezvous point.

Concurrent Number: When the number of users reaches this quantity, the users waiting at the rendezvous are released.

Step 6 When the configuration is complete, click **Save**.

----End

3.5.4.7 Pressure Configuration

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 Click **Pressure Configuration** and set phase parameters by referring to **Table 3-19**. A maximum of 1000 phases can be added to a case.

Table 3-19 Adding a phase

Parameter	Description
Phase Name	Phases names are used to distinguish different phases. For example, you can set the name to **Home page test**.

Parameter	Description
Pressure Mode	The following modes are supported: • **Concurrency Mode**: The performance test is performed based on the number of concurrent users. The number of concurrent users can be a fixed number or can be set to ramp up throughout the phase. • **TPS Mode**: The performance test is performed based on the number of transactions per second (TPS). In TPS mode, each execution of a test case by a virtual user is considered a transaction. The actual TPS may not reach your specified value because it will be affected by the specified maximum number of concurrent users and the average transaction latency. • **Peakload Mode**: The pressure test is performed based on the number of concurrent users who stays at the peak after increasing from the minimum concurrency to the maximum concurrency at a specified ramp-up time. • **Surge Mode**: The pressure test is performed based on the number of concurrent users who changes periodically. In each period, the minimum concurrency is tested first, then the maximum concurrency. The duration is based on the maximum concurrency and can be customized. The number of periods is the specified surge counts. • **Oscillation Mode**: The pressure test is performed based on the number of concurrent users who fluctuates periodically. The number of users in each period fluctuates between the minimum concurrency and the maximum concurrency. The number of periods is the specified oscillation counts. • **Intelligent Peakload**: The pressure test is performed based on the number of concurrent users who stays at the peak after increasing from the minimum concurrency to the maximum concurrency by a specified single-step execution duration. **NOTE** The executor performs a pressure test by gradually increasing the number of concurrent users based on the specified gradient. During the pressure test, the background generates a phase data for each gradient. When the data in a phase has an obvious performance inflection point compared with that in the previous phase, the system determines that the peak performance is reached. The performance inflection point is identified in the following scenarios: The success rate falls below the threshold (100% by default) and the RPS curve shows a downward trend.

Parameter	Description
Execution Policy	This parameter is available only when the concurrency or TPS mode is used. The value can be **Duration** or **Count**. • **Duration**: Tests will be performed based on the specified duration. • **Count**: Tests will be performed based on the total number of tests.
Load Adjustment	This parameter is available only when the concurrency or TPS mode is used, and **Execution Policy** is set to **Duration**. Auto and manual load adjustments are supported. • **Auto**: The maximum number of concurrent users of all stages is used as the maximum concurrency. • **Manual**: The maximum number of concurrent users during a test is used as the maximum concurrency. After manual load adjustments, the original multi-stage pressure configuration becomes invalid.
When **Pressure Mode** is set to **Concurrency Mode**, **Execution Policy** is set to **Duration**, **Load Adjustment** is set to **Auto**, and **Gradient Increment** is disabled, set the following parameters.	
Concurrent Users	Concurrency refers to the number of users performing operations on a system at the same time. In CodeArts PerfTest, it is the number of virtual users you define when you configure a test phase.
Duration (min)	Duration that the phase will last.
When **Pressure Mode** is set to **Concurrency Mode**, **Execution Policy** is set to **Duration**, and **Load Adjustment** is set to **Manual**, set the following parameters.	
Maximum Concurrency	Maximum number of virtual users for a test case.
Start Concurrency	Starting number of virtual users for a test case.
Duration (min)	Duration that the phase will last.
When the pressure mode is set to **Concurrency Mode**, **Execution Policy** is set to **Duration**, **Load Adjustment** is set to **Auto**, and **Gradient Increment** is enabled, set the following parameters. Each case can have only one phase with **Gradient Increment** enabled.	
Start Concurrency	Starting number of concurrent users for a ramp-up.

Parameter	Description
Increasing Concurrency	Total number of concurrent users to be added. During a ramp-up, the starting number of concurrent users is simulated in the first subphase. The number of concurrent users in each subphase is the amount of (**Concurrency** in the previous phase + **Increasing Concurrency** x **Increment**). In the final subphase, the number of concurrent users will reach the amount of (**Start Concurrency** + **Increasing Concurrency**).
Duration per Pressure Test (min)	Duration that each subphase will last.
Increment	Number of concurrent users added in each subphase = **Increasing Concurrency** x **Increment**. The value can be **5%**, **10%**, **20%**, or **50%**.
When **Pressure Mode** is set to **Concurrency Mode** and **Execution Policy** is set to **Count**, set the following parameters. Only a single phase can be added.	
Concurrent Users	The number of concurrent users refers to the number of users performing operations on the system at the same time. In CodeArts PerfTest, it is the number of virtual users you define when you configure a test phase.
Total Sending Times	When the task is running, the performance test in this phase will be executed repeatedly until the number of execution times reaches the specified value. The total sending times must be greater than or equal to the number of concurrent users.
When **Pressure Mode** is set to **TPS Mode**, **Execution Policy** is set to **Duration**, and **Load Adjustment** is set to **Auto**, set the following parameters.	
Maximum Concurrency	Maximum number of virtual users for a test case.
TPS Value	Number of transactions executed per second.
Duration (min)	Duration that the phase will last.
When **Pressure Mode** is set to **TPS Mode**, **Execution Policy** is set to **Duration**, and **Load Adjustment** is set to **Manual**, set the following parameters.	
Maximum Concurrency	Maximum number of virtual users for a test case.
Start Concurrency	Starting number of virtual users for a test case.
TPS Value	Number of transactions executed per second.
Duration (min)	Duration that the phase will last.

Parameter	Description
When **Pressure Mode** is set to **TPS Mode** and **Execution Policy** is set to **Count**, set the following parameters. Only a single phase can be added.	
Maximum Concurrency	Maximum number of virtual users for a test case.
TPS Value	Number of transactions executed per second.
Total Sending Times	When the task is running, the performance test in this phase will be executed repeatedly until the number of execution times reaches the specified value. The total sending times must be greater than or equal to the number of concurrent users.
When **Pressure Mode** is set to **Peakload Mode**, set the following parameters.	
Start Concurrency	Starting number of virtual users for a test case.
Maximum Concurrency	Maximum number of virtual users for a test case.
Ramp Up(s)	Duration for which the number of virtual users increases from the start concurrency to the maximum concurrency. The recommended ramp-up time is shorter than the duration. If the ramp-up time is the same as the duration and the concurrency is large, the sampled maximum concurrency may be slightly less than the configured maximum concurrency.
Duration (min)	Duration that the phase will last.
Expected KPI	• Maximum response time: 60,000 ms • Minimum success rate: 0 In peakload mode, the expected request response time of the tested system is less than or equal to the input value, and the request success rate is greater than or equal to the input value. If an indicator is unacceptable for six consecutive seconds for the first time (for example, the response time is greater than the expected value for six consecutive seconds), use the data in the first second as the analysis result. When the execution of a test case reaches the expected level, the task continues.
When **Pressure Mode** is set to **Surge Mode**, set the following parameters.	
Duration (min)	Duration that the phase will last.
Maximum Concurrency	Maximum number of virtual users for executing a test case, which is also the peak value during the changing number of virtual users.

Parameter	Description
Minimum Concurrency	Minimum number of virtual users for executing a test case, which is also the minimum value during the fluctuation of the number of virtual users.
Surge Count	Number of periods in which the number of virtual users fluctuates within the duration.
Peak Duration(s)	Duration of a stage based on the maximum number of concurrent virtual users in a period.
When **Pressure Mode** is set to **Oscillation Mode**, set the following parameters.	
Duration (min)	Duration that the phase will last.
Maximum Concurrency	Maximum number of virtual users for executing a test case, which is also the peak value during the fluctuation of the number of virtual users.
Minimum Concurrency	Minimum number of virtual users for executing a test case, which is also the trough value during the fluctuation of the number of virtual users.
Oscillation Count	Number of periods in which the number of virtual users fluctuates within the duration.
When **Pressure Mode** is set to **Intelligent Peakload**, set the following parameters.	
Start Concurrency	Starting number of virtual users for a test case.
Increase Concurrency	Number of virtual users increased by step. **Increase Concurrency** increases exponentially.
Single-step Execution Duration(s)	Duration of each incremental concurrency. You are advised to set this parameter to at least 20 seconds.
Duration (min)	Duration that the phase will last. You are advised to set this parameter to less than 30 minutes. When the success rate falls below the threshold (100% by default) and the RPS curve shows a downward trend, the pressure test stops.

Step 5 When the configuration is complete, click **Save**.

----End

3.5.4.8 Advanced Configuration

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 Click **Advanced Configuration**.

1. **Executors**: The setting of the number of executors takes effect only when private resource groups are used. If this parameter is not set, the default policy is used. By default, concurrency supported by a single executor is 5,000 (HTTP/HTTPS), 5,000 (WebSocket), 5,000 (MQTT), 1,000 (JMeter), or 1,000 (HLS/RTMP/HTTP-FLV). Number of executors ≥ Maximum number of concurrent users in all phases/5,000.
2. **Log Collection Policy**: If there are too many requests in your case or cyclic nesting exists, use the request mode.
   - In **Request Mode**, the requests in a case generate logs separately.
   - In **Case Mode**, all requests in a case are associated with each other before logs are generated.

Step 5 When the configuration is complete, click **Save**.

----End

3.5.4.9 SLA Configuration

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 Click **SLA Configuration**. For details, see **Configuring SLAs**.

Step 5 When the configuration is complete, click **Save**.

----End

3.5.5 Setting Global Variables

3.5.5.1 Overview

Global variables are used to build data sets and enrich test data.

If a global variable is used in a packet of a request, variable values in the packet will be replaced with specified values during a pressure test.

Global variables can be used in many scenarios. For example, you can store different usernames and passwords as global variables to simulate a scenario with multiple users.

3.5.5.2 Adding a Global Variable of the Integer or Enumerated Type

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, click **Global Variables**.

Step 4 Add a global variable manually.

1. Click **Add Variable**.
2. Set parameters by referring to **Table 3-20**.

Table 3-20 Setting global variables

Parameter	Description
Name	Name of a new global variable.
Type	Variable type, which can be integer or enumerated.
Value	A pressure test task reads the value range of the corresponding global variable when it is running. For each virtual user, the variable value is polled. That is, the first virtual user obtains the first value of the variable, and the second virtual user obtains the second value. If the values are exhausted, the next virtual user obtains the first value again. Variable content is transmitted in plain text during the pressure test. Exercise caution when entering the content. – When the variable type is **Integer**, enter the value range of the variable. – When the variable type is **Enumerated**, click  to enter a variable value.
Read Mode	**Sequential Mode**: Variables are read in sequence. **Random Mode**: Variables are read randomly.
Sharing Mode	**Case**: All concurrent requests share a variable. **Concurrency**: Each request copies a variable with the same name for use. Different concurrent requests read the variable values in sequence and do not affect each other.

For details about the variable read rules for different variable read and sharing modes, see **Variable Read Rules**.

3. When the settings are complete, click ✓ to save them.
4. After creating a variable, you can perform the following operations:
   - Click **Edit** to modify it. If the global variable is referenced by a transaction, the value of the global variable in the transaction will be changed synchronously after the modification.
   - Click **Delete** to delete the added global variable file. The global variable file cannot be deleted when it is referenced.

----End

3.5.5.3 Adding a .csv or .xlsx Global Variable File

If some parameters are dynamically obtained and the number of parameters is large, you can use .csv or .xlsx files to dynamically transfer the parameters during pressure tests.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, click **Global Variables**.

Step 4 You can add global variables by importing file variables.

1. Click **Download the .csv Template** or **Download the .xlsx Template** to obtain the required variable file template, and enter variables and values in the template. In the templates, enter variable names in the first row and enter variable values from the second row.

   The restrictions on variable files are as follows:

   a. Only .csv (UTF-8 without BOM) and .xlsx files can be imported.
   b. The maximum length of a file name (including the extension) is 50 bytes.
   c. The maximum size of an .xlsx file is 10 MB and that of a .csv file is 80 MB.
2. Click **Import File Variable** to import a .csv or .xlsx file.
   - **Name**: name of the imported file
   - **Type**: **File**
   - **Value**: actual variable name. Multiple variable names are separated by commas (,).
   - **Read Mode**
     - **Sequential Mode**: Variables are read in sequence.
     - **Random Mode**: Variables are read randomly.
   - **Sharing Mode**
     - **Case**: All concurrent requests share a variable.

- **Concurrency**: Each request copies a variable with the same name for use. Different concurrent requests read the variable values in sequence and do not affect each other.

3. After importing a global variable, you can perform the following operations:
   - Click **Edit** to modify the variable read mode and sharing mode.

     For details about the variable read rules for different variable read and sharing modes, see **Variable Read Rules**.
   - Click **Download** to download the global variable file.
   - Click **Delete** to delete the imported global variable file. The global variable file cannot be deleted when it is referenced.
   - If the imported global variable file is modified locally, click **Upload Again** to upload it again.

----End

3.5.5.4 Variable Read Rules

Table 3-21 lists the rules for reading manually specified or imported variables when different variable read modes and sharing modes are selected.

Table 3-21 Variable read rules

Variable Read Mode	Variable Sharing Mode	Description	Example
Sequential	Case	Each concurrency reads a variable value in sequence.	For example, for a task with a concurrency of 10, each concurrency reads a variable value in sequence.
Sequential	Concurrency	Each concurrency copies a variable with the same name for use. Different concurrent requests read the variable values in sequence and do not affect each other.	For example, for a task with a concurrency of 10, each concurrency reads the copied variable values in sequence.
Random	Case	Each concurrency randomly reads a variable value.	For example, for a task with a concurrency of 10, each concurrency randomly reads a variable value.
Random	Concurrency	Each concurrency copies a variable with the same name for use. Different concurrencies read the variable values randomly and do not affect each other.	For example, for a task with a concurrency of 10, each concurrency reads the copied variable values randomly.

Assume that the **number** variable has three values: **1**, **2**, and **3**, and there are two concurrencies: **A** and **B**.

- In sequential mode:

  Case mode: **A** reads **1** of **number**, and **B** reads **2** of **number**. The process goes on cyclically.

  Concurrency mode: **A** reads a copied variable **number** starting from **1**.
  Concurrency mode: **B** also reads a copied variable **number** starting from **1**.

- In random mode:

  The method of reading variable values is the same as that in sequential mode, regardless whether the **Case** or **Concurrency** sharing mode is used.

3.5.5.5 Inserting a Variable

When adding a request, enter **$** in the text box to insert a variable.

When Type is set to Custom, set the following parameters:

- **Range**: variable range.
- **Name**: name of a custom variable. Specifies the name of a variable when the variable is added.

When Type is set to System, select a function name.

- **ID card number**: Randomly generates an ID number.
- **Mobile phone number**: Randomly generates a phone number.
- **Random number in a range**: Randomly generates an integer within the specified range.
- **Random character string**: Randomly generates a string consisting of lowercase letters and numbers based on your specified string length (1 to 32 characters).
- **Time stamp**: Generates a timestamp for the current time based on your specified unit. A 10-digit timestamp represents time in seconds whereas a 13-digit timestamp represents time in milliseconds.
- **UUID**: randomly generates a string of 32 characters.

3.5.6 Binding a Domain Name

CodeArts PerfTest allows you to configure a Domain Name Service (DNS) address pool to map the domain name to the address. The domain name is defined in the URL and the IP address is automatically obtained through the DNS address pool.

- Domain names cannot be bound to video source addresses of RTMP.
- Domain names cannot be bound to MQTT transmission addresses.

Prerequisites

You can bind a domain name only when the **Request URL** on the **Packet** tab page contains the domain name.

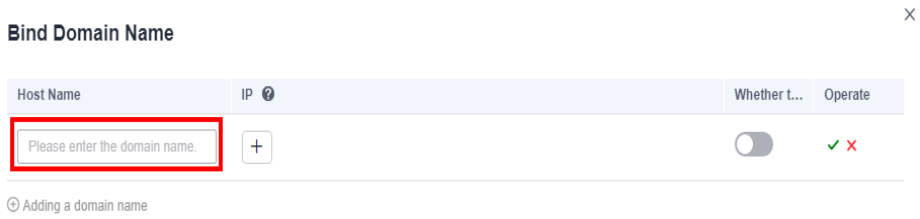
Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab page, click **Domain name binding**

Step 4 Click **Adding a domain name**, and enter the domain name and IP address.



Step 5 Click the enable button, enter an IP address, and click  .

----End

3.5.7 Resetting Configurations

When you modify a test case, you can reset the configurations that are not saved.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the desired test case in the case list on the left.

Step 4 Click  in the upper right corner of the page. In the dialog box that is displayed, click **OK**.

----End

3.5.8 Managing Test Cases

Copying a Test Case

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

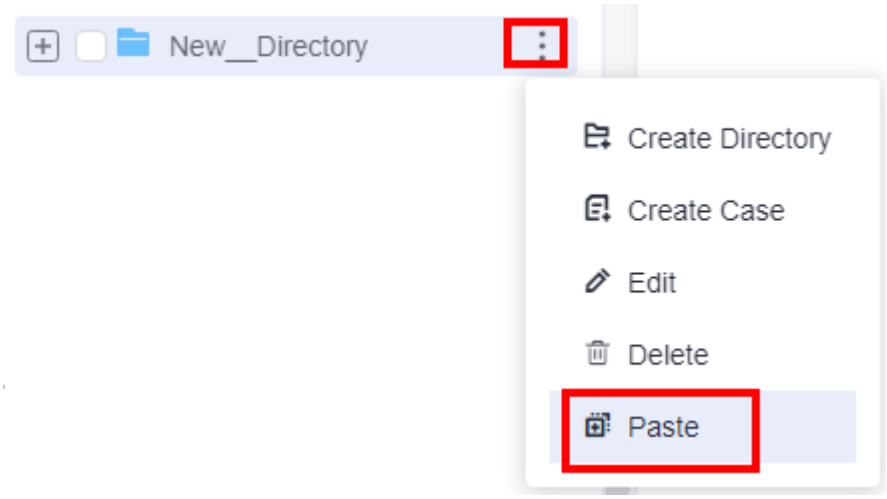
Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, locate the case to be copied in the case list on the left.

Step 4 Click next to the case, and choose **Copy** from the pop-up menu.

Step 5 Click next to the destination directory, and choose **Paste** from the pop-up menu.

Figure 3-10 Pasting a case



Step 6 Click **Save**.

----End

Deleting a Test Case

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, locate the case to be deleted in the case list on the left.

Step 4 Click next to the case, and choose **Delete** from the pop-up menu.

Step 5 Click **OK**.

----End

3.5.9 Debugging a Case

Prerequisites

- Test cases have been added.
- The resource group is in the **Running** state.
- Ports 32001 and 32003 on the debugging node of the resource group are enabled in the security group.
- The execution node of the resource group and applications under the pressure test are connected through the network.

Debugging a Test Case

After you create or edit a test case, you can debug it to detect syntax or configuration errors before testing.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select the test case to be debugged in the case list on the left.

Step 4 Click **Debug** in the upper right corner of the page.

Step 5 Select the target test resource group as the executor and click **Start** to start debugging.

Step 6 Click **View log** to view the test case debugging details.

If an error was reported in the debugging result, edit the case based on the log information and debug it again.

Step 7 On the **Debug log** tab page, you can view the debugging history.

----End

3.5.10 Batch Operations

The case list supports batch operations. Batch operations apply only to cases (including cases in the directory but excluding the directory itself).

Batch Starting

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select multiple cases or directories in the case list on the left.

Step 4 In the floating box displayed in the lower part of the page, click **Start**.

Step 5 In the displayed **Executing Temporary Test Task** dialog box, modify the parameters of the temporary task.

Table 3-22 Temporary test tasks

Parameter	Description
Temporary Task Name	The current case will not generate a test task, and the generated test report will be displayed in the performance report as a temporary task report and automatically deleted 30 days later.
Execution Mode	Serial or parallel

Parameter	Description
Resource Group Type	Shared resource group or private resource group

Step 6 Click **Start** to execute the temporary task.

Step 7 After the test case is started, you can click **View Report** to view the real-time performance report.

You can also switch the tab page in the upper left corner and choose **Performance Reports** > **Temporary Task List** to view the performance report by temporary task name.

----End

Batch Deleting Test Cases

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Cases** tab, select multiple cases or directories in the case list on the left.

Step 4 In the floating box displayed in the lower part of the page, click **Delete**.

Step 5 Click **OK**.

----End

3.6 PerfTest Task Management

3.6.1 Creating a Test Task

A test task initiates a performance test based on a defined test model. It is a series of tests performed at different pressure points to continuously initiate pressure tests on a system. Testing allows you to obtain and analyze the performance data of the system. You can add multiple test tasks to a test project.

Prerequisites

- A test case has been created and configured based on service requirements.
- (Optional) A transaction has been added.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Test Tasks** tab, click **Create Task**.

Step 4 Enter a task name and select an execution policy.

There are two execution policies:

- **Serial**: Test cases in the test task are executed in sequence.
- **Parallel**: Test cases in the test task are executed concurrently.

Step 5 Click **Add Case**. In the dialog box that is displayed, select created cases.

A test case can be added to multiple test tasks.

Step 6 When the configuration is complete, click **Save**.

----End

3.6.2 Starting a Test Task

Prerequisites

- Test cases have been added.
- The resource group is in the **Running** state.
- Ports 32001 and 32003 on the debugging node of the resource group are enabled in the security group.
- The execution node of the resource group and applications under the pressure test are connected through the network.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 Click the **Test Tasks** tab, and then click ▶ in the **Operation** column of the desired task.

Step 4 In the **Executing Test Task** dialog box, select a resource group type from the **Resource Group Type** drop-down list.

Step 5 Click **Start** to start the test task.

Step 6 (Optional) After the test task is executed, click **View Report** or  in the **Operation** column of the test task to view the real-time test report.

----End

3.6.3 Managing Test Tasks

After a test task is created, you can manage it and the task phases.

Starting Test Tasks in Batches

Start multiple test tasks under the same test project.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Test Tasks** tab page, select multiple tasks and click **Start**.

Step 4 Select a resource group.

Step 5 Click **Start**.

----End

Stopping Test Tasks

Stop multiple test tasks under the same test project.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Test Tasks** tab, click ■ in the row containing the desired test task. You can also select multiple tasks and click **Stop** to stop them at a time.

----End

Editing a Test Task

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 Click the **Test Tasks** tab, and then click ✎ in the **Operation** column of the desired task.

Step 4 Edit the test task name, execution policy, and cases as required.

- Click the original task name to display a text box. Enter a new task name in the box.
- Modify the execution policy.
- Delete or add test cases.

Step 5 Click **Save**.

----End

Deleting a Test Task

Deleted test tasks cannot be recovered. Exercise caution when deleting a task.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired PerfTest project to go to the details page.

Step 3 On the **Test Tasks** tab, click 🗑 in the row containing the desired test task.

You can select multiple test tasks and click **Delete** to delete them at a time.

Step 4 Delete the test tasks as prompted.

----End

3.7 PerfTest Report Management

3.7.1 Test Report Description

Report Overview

For details about the test report, see **Table 3-23**.

This report shows the response performance of the tested system in a scenario with a large number of concurrent users. To understand the report, see the following information:

- **Statistical dimension**: In this report, RPS, response time, and concurrency are measured in a single case. If a request has multiple packets, the request is considered successful only when all the packets are responded to. The response time of the request is the sum of the response time of the packets.
- **Response timeout**: If the corresponding TCP connection does not return the response data within the set response timeout (5s by default), the case request is counted as a response timeout. Possible causes include: the tested server is busy or crashes, or the network bandwidth is fully occupied.
- **Verification failure**: The response packet content and response code returned from the server do not meet the expectation (the default expected response code of HTTP/HTTPS is 200), such as code 404 or 502. A possible cause is that the tested service cannot be processed normally in scenarios with a large number of concurrent users. For example, a database bottleneck exists in the distributed system or the backend application returns an error.
- **Parsing failure**: All response packets are received, but some packets are lost. As a result, the entire case response is incomplete. This may be caused by network packet loss.
- **Bandwidth**: This report collects statistics on the bandwidth of the execution end of CodeArts PerfTest. The uplink indicates the traffic sent from CodeArts PerfTest, and the downlink indicates the received traffic. In the external pressure test scenario, you need to check whether the EIP bandwidth of the executor meets the uplink bandwidth requirement and whether the bandwidth exceeds the downlink bandwidth of 1 GB.
- **Requests per second (RPS)**: indicates the number of requests per second. Average RPS = Total number of requests in a statistical period/Statistical period.
- **SLA Result:** If you have configured SLA rules for a test case, you can view the SLA report to check whether the SLA rules are triggered during testing. In the SLA report, **Request Name** is the name of the request in the test case. **Rules** indicates the configured SLA rules. **Average Value** (displayed as the current value for a running task) indicates the average value of service metrics and the total number of times that SLA rules have been triggered since the pressure test starts. **Trigger Events** displays the request name and when and how many times the SLA rules were triggered during testing.

- **How to determine the quality of tested applications**: According to the service quality definition of an application, the optimal status is that there is no response or verification failure. If there is any response or verification failure, it must be within the defined service quality range. Generally, the value does not exceed 1%. The shorter the response time is, the better. User experience is considered good if the response time is within 2s, acceptable if it is within 5s and needs optimization when it is over 5s. TP90 and TP99 can objectively reflect the response time experienced by 90% to 99% of users.

Table 3-23 Test report description

Parameter	Description
Total Number of Metrics	Total number of metrics of all cases. • **Maximum Concurrency**: indicates the maximum number of concurrent virtual users. • **RPS**: requests per second. • **Response Time**: duration from the time when a client sends a request to the time when the client receives a response from the server. • **Response Code**: distribution of response codes during the running of a pressure test task. • **Bandwidth**: records the real-time bandwidth usage during the running of a pressure test task. • **SLA Events**: occurrence of an event defined in the SLA. • **Normal Response**: indicates the number of case responses that pass the set checkpoints. If no checkpoints are set, the number of case responses with code 2XX will be calculated by default. • **Abnormal Response**: indicates the number of parsing failures, verification failures, response timeout, 3XX, 4XX, 5XX, and connection failures. • **Success Rate**: Number of requests that are normally returned/Total number of requests.
Average RPS	Total number of requests in a statistical period/Statistical period
Average RT	Average response time of all requests sent in a second
Concurrent Users	Changes on the number of concurrent virtual users during testing

Parameter	Description
Bandwidth (kbit/s)	Records the real-time bandwidth usage during the running of a pressure test task. • **Uplink bandwidth**: speed at which the CodeArts PerfTest execution node sends out data. • **Downlink bandwidth**: speed at which the CodeArts PerfTest execution node receives data In the original data exported from the test report page, the uplink and downlink bandwidths are average values. The uplink bandwidth is equal to the total uplink bandwidth divided by the number of requests, and the downlink bandwidth is the total downlink bandwidth divided by the number of requests.
Response distribution statistics	Number of cases processed per second for normal return, parsing failure, verification failure, response timeout, connection rejection, and other errors. This metric is related to the think time, concurrent users, and server response capability. For example, if the think time is 500 ms and the response time of the last request for the current user is less than 500 ms, the user requests twice per second. • **Normal Response**: indicates the number of case responses that pass the set checkpoints. If no checkpoints are set, the number of case responses with code 2XX will be calculated by default. • **Abnormal Response**: indicates the number of parsing failures, verification failures, response timeout, 3XX, 4XX, 5XX, and connection failures. • **Parsing Failure**: indicates the number of HTTP responses that fail to be parsed. • **Verification Failure**: indicates the number of case responses that do not pass the set checkpoints. If no checkpoints are set, the number of case responses returning contents other than 2XX will be calculated by default. • **Response Timeout**: indicates the number of case requests that do not receive responses from the server within 5 seconds after a request packet is sent. • **Rejected**: indicates the number of rejected connection requests. • **Others**: indicates the number of other errors. **NOTE** Sampling algorithm is applied to the curve data, leading to possible data distortion on the test report page. The details are as follows: When the case execution duration is longer than 15 minutes, the curve data on the GUI samples instantaneous values by proportion. The data at the time points that are not sampled is not displayed on the GUI.

Parameter	Description
Response Code statistics	1XX/2XX/3XX/4XX/5XX
Response Time Ratio	Ratio of response time of cases.
Maximum Response Time of TP (ms)	If you want to calculate the top percentile *XX* (TP*XX*) for a request, collect all the response time values for the request over a time period (such as 10s) and sort them in an ascending order. Remove the top (100–*XX*)% from the list, and the highest value left is the value of TP*XX*. • TP50: Remove the top 50% from the list, and the highest value left is the value of TP50. • TP75: Remove the top 25% from the list, and the highest value left is the value of TP75. • TP90: Remove the top 10% from the list, and the highest value left is the value of TP90. • TP95: Remove the top 5% from the list, and the highest value left is the value of TP95. • TP99: Remove the top 1% from the list, and the highest value left is the value of TP99. • TP99.9: Remove the top 0.1% from the list, and the highest value left is the value of TP99.9. • TP99.99: Remove the top 0.01% from the list, and the highest value left is the value of TP99.99.

NOTE

Streaming protocol requests do not contain data such as the response time and response code. Therefore, in the test report of this type of tasks, the values of related data metrics (including the average RT, response code distribution, response time range proportion, and maximum TP response time) are 0.

3.7.2 Viewing a Real-Time Test Report

You can view the monitored metrics during a performance test in a real-time test report.

Prerequisites

A test task has been started.

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **PerfTest Projects** in the left navigation pane, and choose > **View Report** in the row containing the desired test project.

Step 2 On the **Performance Reports** tab page, locate the desired task. The task must be running.

Step 3 Click the task name or click  in the **Operation** column to view the real-time test report. Click **Stop Task** in the upper right corner of the page to stop the task. For details about the parameters, see **Test Report Description**.

Step 4 On the **Overview** tab page, you can view the number of failed/total requests, average latency, RPS, maximum concurrency, success rate, SLA alarms, bandwidth, dynamic trend, and response codes.

Step 5 On the **Detail** tab page, you can view the logs, general test metrics, request details, and the SLA result of the test case.

1. Click **View Log**. In the displayed dialog box, click Download Request Log to obtain the request log table. **Table 3-24** describes the table headings in the downloaded table. Ten request logs are displayed based on the name, return code, and result. There are no more than two request log files.

   Table 3-24 Description of fields in the table header

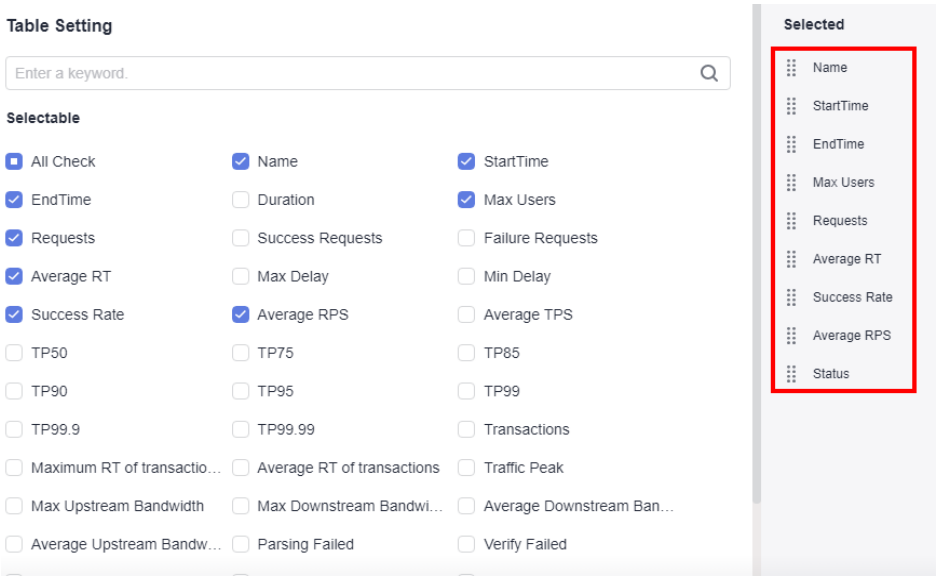
Table Header Field	Description
Name	Name of a request
Result	Return result of the request. The value can be **FAILURE** or **SUCCESS**.
Begin Time	Start time of a request
End Time	End time of a request
URL	URL of a request
Method	Method for invoking a request
Request Header	Request header information
Request Body	Request body information
Response Code	Response code
Response Header	Response header information
Response Body	Response body information
Failure Message	Cause of a failed request

2. During the execution of a test task, you can click **Process Voltage Adjustment** to dynamically adjust the pressure based on the real-time data. For the pressure mode, only the concurrency mode, TPS mode, and duration policy are supported.
3. The sampling modes displayed in the report details are equidistant sampling value and equidistant average value.

- Equidistant sampling value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with sampling points at a fixed interval.
- Equidistant average value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with average values at a fixed interval.

4. Click the **Data View** drop-down list box and enter a keyword to search for the required case or request data. You can also click the case or request data to be displayed in the case directory in the **Data View** drop-down list box.
5. On the **Detail** tab page, you can also click **List** to go to the report metric summary page.
   - Click **Customized Column**. In the displayed dialog box, select the items to be displayed, and drag the selected items in the **Selected** list on the right to change the item sequence.

     Figure 3-11 Table settings

   - Click  in the **Operation** column to view logs.
   - Click  in the **Operation** column to adjust the process voltage. For the pressure mode, only the concurrency mode, TPS mode, and duration policy are supported.
   - Click  in the **Operation** column to edit a test case.

----End

3.7.3 Viewing an Offline Test Report

After a pressure test is complete, the system generates an offline test result report.

Prerequisites

The test task is complete.

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **PerfTest Projects** in the left navigation pane, and choose > **View Report** in the row containing the desired test project.

Step 2 On the **Performance Reports** tab page, select a task to view its test report.

Step 3 Click the task name, or click in the **Operation** column. In the **Report List** on the left, select the offline report to be viewed. For details, see **Table 3-23**. CodeArts PerfTest retains offline report data for three years. You can download an offline PDF report to a local PC and export the raw data (in CSV) for further processing.

Step 4 On the **Overview** tab page, you can view the number of failed/total requests, average latency, RPS, maximum concurrency, success rate, SLA alarms, bandwidth, dynamic trend, and response codes.

Step 5 On the **Detail** tab page, you can view the logs, general test metrics, request details, and the SLA result of the test case.

1. Click **View Log**. In the displayed dialog box, click Download Request Log to obtain the request log table. **Table 3-25** describes the table headings in the downloaded table. Ten request logs are displayed based on the name, return code, and result. There are no more than two request log files.

   Table 3-25 Description of fields in the table header

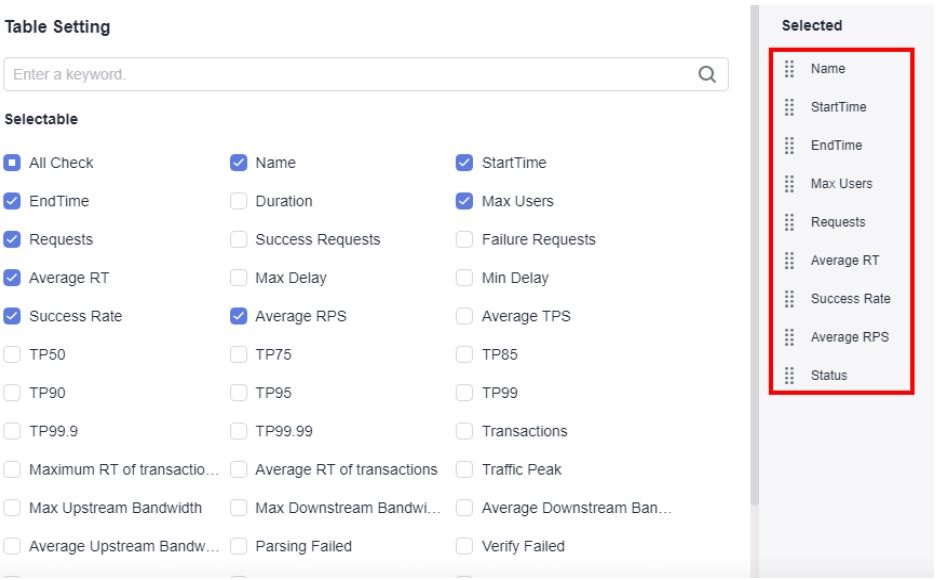
Table Header Field	Description
Name	Name of a request
Result	Return result of the request. The value can be **FAILURE** or **SUCCESS**.
Begin Time	Start time of a request
End Time	End time of a request
URL	URL of a request
Method	Method for invoking a request
Request Header	Request header information
Request Body	Request body information
Response Code	Response code
Response Header	Response header information
Response Body	Response body information
Failure Message	Cause of a failed request

2. The sampling modes displayed in the report details are equidistant sampling value and equidistant average value.

- Equidistant sampling value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with sampling points at a fixed interval.
- Equidistant average value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with average values at a fixed interval.

3. Click the **Data View** drop-down list box and enter a keyword to search for the required case or request data. You can also click the case or request data to be displayed in the case directory in the **Data View** drop-down list box.
4. On the **Detail** tab page, you can also click **List** to go to the report metric summary page.
   - Click **Customized Column**. In the displayed dialog box, select the items to be displayed, and drag the selected items in the **Selected** list on the right to change the item sequence.

     Figure 3-12 Table settings



   - Click  in the **Operation** column to view logs.
   - Click  in the **Operation** column to edit a test case.

----End

3.7.4 Report Comparison

For the test reports generated at different time or under different conditions, you can view the test result by comparing the reports.

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **PerfTest Projects** in the left navigation pane, and choose  > **View Report** in the row containing the desired test project.

Step 2 On the **Performance Reports** tab page, locate the desired task, and click the task name.

Step 3 In the **Report List** area, click **Comparison**.

Step 4 Select desired test reports and click **OK**.

You can select a maximum of three offline reports for comparison. The first selected report is used as the baseline report.

Step 5 Select a case from the **Test Case Comparison** box to compare the metrics of this case in different reports.

----End

3.8 Transaction Management

3.8.1 Adding a Transaction

A transaction refers to a complete operation process from end to end, such as a login, criteria-based query, and payment. A transaction can be used by multiple test cases. CodeArts PerfTest supports flexible combination of transactions. You can add multiple transactions to a test project.

Prerequisites

A test project has been created. For details on how to create a test project, see **Creating a Test Project**.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transaction Library** tab page, click  on the left of the page to add a transaction, configure the following parameters, and click **OK**.

- **Transaction Name**: name of a customized transaction.
- **Type**: Normal transaction refers to transactions of performance tests in normal scenarios, including API tests of applications built based on HTTP/HTTPS/TCP/UDP/HLS/RTMP/WebSocket/HTTP-FLV/MQTT protocols. Transaction requests can be connected with each other.

Step 4 Add requests to the transaction and click **OK**.

When the transaction type is **Normal**, you can add four types of request compositions: packet, think time, response extraction, and checkpoint. Packet is mandatory.

- **Packet**: data blocks transmitted between HTTP-based applications. For details, see **Adding Request Information (Packet)**.

- **Think Time**: time interval between two actions. For details, see **Adding Request Information (Think Time)**.
- **Response Extraction**: If multiple packets exist in the same transaction, use regular expressions or JSON to extract the output of the previous packet for the input of the next packet. For details, see **Adding Request Information (Response Extraction)**.
- **Checkpoint**: used to verify if contents returned by servers are correct through customized verification information. For details, see **Adding Request Information (Checkpoint)**.

Step 5 (Optional) A transaction contains one or more requests. If your transaction contains multiple requests, click **Add Request** to add more requests. Up to 40 requests can be added to a normal transaction.

----End

3.8.2 Debugging Transactions

After adding or modifying a transaction, you can quickly find syntax or configuration errors through debugging.

Prerequisites

- The resource group is in the **Running** state.
- Ports 32001 and 32002 on the debugging node of the resource group are enabled in the security group.
- The network between the debugging node of the resource group and the tested application is normal.

Procedure

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transactions** tab page, click **Debug** next to the transaction to be debugged.

Step 4 Select the target test resource group as the executor and click **OK**.

Step 5 Click **View log** to view the transaction debugging details.

Step 6 On the **Debug log** tab page, you can view the debugging history.

----End

3.8.3 Managing Transactions

After a transaction is created, you can modify, delete, and copy the transaction.

A transaction used by a test cannot be deleted.

Modifying a Transaction

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transactions** tab page, select the desired transaction in the left pane and click **Edit** in the upper part of the list.

Step 4 Modify the transaction name and click **OK**.

----End

Deleting a Transaction

Deleted transactions cannot be restored. Exercise caution when deleting a transaction.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transactions** tab page, select the desired transaction in the left pane, click **Delete** in the upper part of the list, and delete the transaction as prompted.

You can select multiple transactions and click **Delete** to delete these transactions at a time.

----End

Copying a Transaction

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transactions** tab page, select the desired transaction in the left pane, click **Copy** in the upper part of the list, and click **OK**.

Step 4 Enter the name of the new transaction and click **OK**.

----End

3.8.4 Managing Transaction Request Information

After a transaction request is created, you can insert, modify or delete it.

Inserting Request Information

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transactions** tab page, select the transaction request to be inserted, click in the upper right corner, and click **Insert a request**.

Step 4 Configure the request to be inserted and then click **OK**.

----End

Modifying a Transaction Request

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transactions** tab page, select the transaction whose request information is to be modified, click  next to the desired request, and modify the request information as required. Then click **OK**.

----End

Disabling/Enabling a Transaction Request

If you want to retain a request of a transaction but do not want to use it, you can disable it. If you want to resume the use of the request, you can enable it again.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 Disable a transaction request.

On the **Transactions** tab page, select the transaction whose request is to be disabled, click  next to the desired request to disable it. Other requests are not affected.

Step 4 Enable a transaction request.

On the **Transactions** tab page, select the transaction whose request is to be enabled, click  next to the desired request to enable it. Other requests are not affected.

----End

Deleting a Transaction Request

Deleted transaction requests cannot be restored. Exercise caution when deleting a transaction request.

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Locate a PerfTest project to be edited and click  to edit the transaction library.

Step 3 On the **Transactions** tab page, select the transaction whose request is to be deleted, click  next to the desired request to delete it. Other requests are not affected.

----End

3.9 JMeter Test Project Management

3.9.1 Managing JMeter Projects

A JMeter test project provides the native JMeter engine. You can import JMeter scripts into JMeter test projects, use the native JMeter engine to quickly initiate high-concurrency performance tests, and view complete performance test reports.

Creating a JMeter Project

Step 1 Log in to the CodeArts PerfTest console, choose **JMeter Projects** in the left navigation pane, and click **Create JMeter Project**.

Step 2 Set the basic information by referring to **Table 3-26**.

Table 3-26 Creating a JMeter project

Parameter	Description
Project Name	Name of a new test project. The project name can contain a maximum of 128 bytes, including letters, digits, hyphens (-), underscores (_), and periods (.).
Description	Description of a new test project.

Step 3 When the configuration is complete, click **OK**.

When the JMeter project is created, you can add a test plan for it. For details, see **Managing JMeter Test Plans**.

----End

Renaming and Deleting a Test Project

CodeArts PerfTest allows you to modify and delete a created JMeter project.

Step 1 Log in to the CodeArts PerfTest console and choose **JMeter Projects** in the left navigation pane.

Step 2 In the JMeter test project list, click > **Modify Project** in the row containing the desired project. Enter a new project name and click **OK**.

Step 3 In the JMeter test project list, click > **Delete Project** in the row containing the desired project. Delete the project as prompted.

----End

3.9.2 Managing JMeter Test Plans

Creating a Test Plan

A test plan is used to initiate a performance test based on the defined JMeter files, JAR packages, and variable files.

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click in the row containing the desired project.

Step 2 On the **Test Plans** tab, click **Add Task Plan**.

Step 3 In the dialog box that is displayed, click **Select File**, select a .jmx file, and click **Upload**. When the file is imported, the dialog box closes automatically and the **Test Plans** tab is displayed. You can view the added test plan.

- The name (including the extension) of a JMX file can contain up to 64 bytes. The file size can be up to 10 MB.
- Basic JMeter scripts are supported, including thread group, HTTP request, HTTP Header Manager, request parameters, and timeout interval. You are advised to use a script of version 5.2.1 or later. Otherwise, a script parsing error may occur.

Step 4 On the **Test Plans** tab page, click **Third Party Jar**. In the displayed dialog box, click **Select File**, select and import the JAR package on which the test plan depends. After the package is imported, close the dialog box.

- Java 9 and later versions do not support the upload of third-party JAR packages.
- The name (including the extension) of a JAR package can contain up to 64 bytes. The package size can be up to 10 MB.

Step 5 View the newly added test plan on the **Test Plans** tab.

----End

Pressure Configuration

After creating a test plan, you can edit phase parameters.

Step 1 Log in to the CodeArts PerfTest console and choose **JMeter Projects** in the left navigation pane. Click in the row of the test project to which the desired test plan belongs.

Step 2 On the **Test Plans** tab page, click the name of the desired test plan. On the **Plan Detail** page, you can view the stages of different thread groups. Set the basic information by referring to **Table 3-27**.

Table 3-27 Pressure configuration

Parameter	Description
Execution Policy	The value can be **Duration** or **Count**. • **Duration**: Tests will be performed based on the specified duration. • **Count**: Tests will be performed based on the total number of tests.
Number of Threads	Number of concurrent virtual users.
Ramp-up Period	This parameter is involved only in **Phase1**. It corresponds to the ramp-up period in JMeter scripts. The parameter indicates the time required for increasing the number of concurrent users from 0 to the specified value.
Loop Count	This parameter is available only when **Execution Policy** is set to **Count**. Number of execution times of each virtual user.
Specify Thread Lifetime	This parameter is available only when **Execution Policy** is set to **Count**. If you enable this option, you can configure the **Duration** and **Startup Delay** parameters.
Duration	This parameter is available only when **Execution Policy** is set to **Count** and **Specify Thread Lifetime** is enabled. If the actual test duration reaches the maximum duration and the number of execution times does not reach the number of cycles, the thread group execution ends.
Duration	This parameter is available only when **Execution Policy** is set to **Duration**. How long the test will run. It is recommended that the duration be set to at least 300 seconds.
Startup Delay	When a test task is started, CodeArts PerfTest will wait the duration of startup delay before generating virtual users. This parameter can be configured when **Execution Policy** is set to **Count** and **Specify Thread Lifetime** is enabled. When **Execution Policy** is set to **Duration**, you can modify this parameter for **Phase1**.

Step 3 Click **OK**.

----End

Debugging a Test Plan

When a test plan is added or modified, you can debug it to quickly detect syntax or configuration errors and ensure that the model is available in a performance test.

Step 1 Log in to the CodeArts PerfTest console and choose **JMeter Projects** in the left navigation pane. Click  in the row of the test project to which the desired test plan belongs.

Step 2 On the **Test Plans** tab page, click the name of the task to be debugged. On the plan details page that is displayed, click  in the upper part of the page.

Step 3 On the debugging page, select a created private resource group as the executor and click **Start** to start debugging. You can view the debugging progress on the debugging page.

Step 4 After the debugging is complete, if an error is reported, modify the case based on the exception information and debug the case again. For details on how to import a .jmx file, see the section on modifying a test plan in **Managing Test Plans**.

Step 5 Click **View log** to view the test plan debugging details.

Step 6 On the Debug log tab page, you can view the debugging history.

----End

Starting a Test Plan

Test plans are a series of tests performed at different pressure points to continuously initiate pressure tests on a system. Testing allows you to obtain and analyze the performance data of the system.

You can add multiple test plans to a test project.

Prerequisites

- The resource group is in the **Running** state.
- Ports 32001 and 32003 on the debugging node of the resource group are enabled in the security group.
- The execution node of the resource group and applications under the pressure test are connected through the network.

Procedure

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click  in the row containing the desired project.

Step 2 Create a test plan. For details, see **Creating a Test Plan**.

Step 3 Click  in the row containing the created task.

Step 4 In the **Start Test Task** dialog box, select a resource group. Only private resource groups are supported for executing JMeter tasks.

Step 5 Click **Start** to start the test task. Click **View Report** to view the real-time performance report.

It is recommended that the duration of a pressure test be at least 300s and the number of concurrent users be set based on the actual situation. Obtain the maximum pressure value for the application through repeated tests by adjusting test data, and perform continuous optimization and verification for the application.

----End

Managing Test Plans

You can manage created test plans.

Enabling test plans in batches

Enable multiple test plans in a test project.

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click  in the row containing the desired project.

Step 2 On the **Test Plans** tab page, select multiple test plans and click **Start**.

Step 3 Select a resource group and click **Start**.

----End

Advanced configuration of test plans

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click  in the row containing the desired project.

Step 2 On the **Test Plans** tab page, click  in the row containing the desired plan.

Step 3 Configure advanced parameters.

1. Executor: The number of executors takes effect only when private resource groups are used. If this parameter is not set, the default policy is used, that is, a single executor supports concurrent JMeter (1000). Number of executors ≥ Maximum number of concurrent users in all phases/1,000.
2. Failure Log Collection Ratio: By default, the sampling ratio is 10‰, and can be set to 1000‰ at most.

Step 4 When the configuration is complete, click **OK**.

----End

Editing a test plan

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click  in the row containing the desired project.

Step 2 On the **Test Plans** tab page, click **More** in the **Operation** column of the desired plan, and choose **Update jmx**.

Step 3 In the displayed **Edit Test Plan** dialog box, import a new .jmx file, and then click **Close**.

Step 4 On the **Test Plans** tab page, click **More** in the **Operation** column of the desired plan, and choose **Variable File** to import the files to be referenced by the test plan.

The file restrictions are as follows:

1. Only .csv (UTF-8 without BOM) and .xlsx files can be imported.
2. The maximum length of a file name (including the extension) is 64 bytes.
3. The maximum size of an .xlsx file is 10 MB and that of a .csv file is 80 MB.

----End

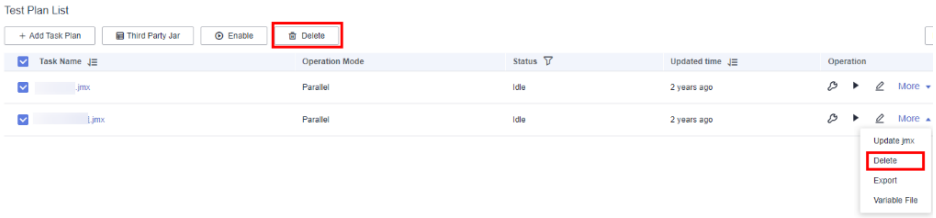
Deleting a test plan

Deleted test plans cannot be recovered. Exercise caution when deleting a test plan.

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click in the row containing the desired project.

Step 2 On the **Test Plans** tab page, click **More** in the **Operation** column of the desired plan, and choose **Delete**. You can also select multiple test plans and click **Delete** to delete them all.

Figure 3-13 Deleting one or more test plans



Step 3 Delete the test plans as prompted.

----End

Deleting phases

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click in the row containing the desired project.

Step 2 On the **Test Plans** tab page, click the task name to go to the **Plan Detail** page.

Step 3 Select the desired thread group, click next to the phase to be deleted, and click **OK**.

Phase1 cannot be deleted.

----End

Exporting JMX Files, Variable Files, and JAR Packages

Prerequisites: The JMX files, variable files, and JAR packages have been imported.

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **JMeter Projects**. Then click in the row containing the desired project.

Step 2 On the **Test Plans** tab page, click **More** in the **Operation** column of the desired plan, and choose **Export** to export its .jmx file.

Step 3 Click **More** in the row containing the test plan whose variable file is to be exported and choose **Variable File**. The **Variable File** dialog box is displayed.

Step 4 Click **Export** in the row containing the variable file to be exported.

Step 5 Click **Third Party Jar**. The **Third Party JAR Package** dialog box is displayed.

Step 6 Click **Export** in the row containing the desired JAR package to export it.

----End

3.9.3 Managing JMeter Test Reports

Test Report Description

Real-time and offline JMeter test reports allow you to view and analyze test data at any time.

For details about the JMeter test report, see **Table 3-28**.

This report shows the response performance of the tested system in a scenario with a large number of concurrent users. To understand the report, see the following information:

- **Statistical dimension**: In this report, RPS, response time, and concurrency are measured in a single thread group. If a request has multiple packets, the request is considered successful only when all the packets are responded to. The response time of the request is the sum of the response time of the packets.
- **Response timeout**: If the corresponding TCP connection does not return the response data within the set response timeout (customized in *.jmx files), the thread group request is counted as a response timeout. Possible causes include: the tested server is busy or crashes, or the network bandwidth is fully occupied.
- **Verification failure**: The response packet content and response code returned from the server do not meet the expectation (the default expected response code of HTTP/HTTPS is 200), such as code 404 or 502. A possible cause is that the tested service cannot be processed normally in scenarios with a large number of concurrent users. For example, a database bottleneck exists in the distributed system or the backend application returns an error.
- **Parsing failure**: All response packets are received, but some packets are lost. As a result, the entire case response is incomplete. This may be caused by network packet loss.
- **Bandwidth**: This report collects statistics on the bandwidth of the execution end of CodeArts PerfTest. The uplink indicates the traffic sent from CodeArts PerfTest, and the downlink indicates the received traffic. In the external

pressure test scenario, you need to check whether the EIP bandwidth of the executor meets the uplink bandwidth requirement and whether the bandwidth exceeds the downlink bandwidth of 1 GB.

- **Requests per second (RPS)**: indicates the number of requests per second. Average RPS = Total number of requests in a statistical period/Statistical period.
- **How to determine the quality of tested applications**: According to the service quality definition of an application, the optimal status is that there is no response or verification failure. If there is any response or verification failure, it must be within the defined service quality range. Generally, the value does not exceed 1%. The shorter the response time is, the better. User experience is considered good if the response time is within 2s, acceptable if it is within 5s and needs optimization when it is over 5s. TP90 and TP99 can objectively reflect the response time experienced by 90% to 99% of users.

Table 3-28 Description of the JMeter test report

Parameter	Description
Total Number of Metrics	Total number of metrics in all thread groups. • **Maximum Concurrency**: indicates the maximum number of concurrent virtual users. • **RPS**: requests per second. • **Normal Response**: indicates the number of transaction responses that pass the set checkpoints. If no checkpoints are set, the number of transactions that return 2XX response code will be calculated by default. • **Response Time**: duration from the time when a client sends a request to the time when the client receives a response from the server. • **Response Code**: distribution of response codes during the running of a pressure test task. • **Bandwidth**: records the real-time bandwidth usage during the running of a pressure test task. • **Abnormal Response**: indicates the number of parsing failures, verification failures, response timeout, 3XX, 4XX, 5XX, and connection failures. • **Success Rate**: Number of requests that are normally returned/Total number of requests.
Average RPS	Total number of requests in a statistical period/Statistical period
Average RT	Changes of the average response time of a pressure test task
Concurrent Users	Changes on the number of concurrent virtual users during testing

Parameter	Description
Bandwidth (kbit/s)	Records the real-time bandwidth usage during the running of a pressure test task. • **Uplink bandwidth**: speed at which the JMeter execution node sends out data. • **Downlink bandwidth**: speed at which the JMeter execution node receives data.
Response distribution statistics	Number of transactions processed per second for normal return, parsing failure, verification failure, response timeout, connection rejection, and other errors. This metric is related to the think time, concurrent users, and server response capability. For example, if the think time is 500 ms and the response time of the last request for the current user is less than 500 ms, the user requests twice per second. • **Normal Response**: indicates the number of transaction responses that pass the set checkpoints. If no checkpoints are set, the number of transactions that return 2XX response code will be calculated by default. • **Parsing Failure**: indicates the number of HTTP responses that fail to be parsed. • **Verification Failure**: indicates the number of transaction responses that do not pass the set checkpoints. If no checkpoints are set, the number of transactions that return 2XX response code will not be calculated. • **Response Timeout**: number of case requests with no responses in the TCP connection within the configured response timeout interval. • **Rejected**: indicates the number of rejected connection requests. • **Others**: indicates the number of other errors. **NOTE** Sampling algorithm is applied to the curve data, leading to possible data distortion on the test report page. The details are as follows: When the case execution duration is longer than 15 minutes, the curve data on the GUI samples instantaneous values by proportion. The data at the time points that are not sampled is not displayed on the GUI.
Response Code statistics	1XX/2XX/3XX/4XX/5XX
Response Time Ratio	Ratio of response time of cases.

Parameter	Description
Maximum Response Time of TP (ms)	If you want to calculate the top percentile *XX* (TP*XX*) for a request, collect all the response time values for the request over a time period (such as 10s) and sort them in an ascending order. Remove the top (100–*XX*)% from the list, and the highest value left is the value of TP*XX*. • TP50: Remove the top 50% from the list, and the highest value left is the value of TP50. • TP75: Remove the top 25% from the list, and the highest value left is the value of TP75. • TP90: Remove the top 10% from the list, and the highest value left is the value of TP90. • TP95: Remove the top 5% from the list, and the highest value left is the value of TP95. • TP99: Remove the top 1% from the list, and the highest value left is the value of TP99. • TP99.9: Remove the top 0.1% from the list, and the highest value left is the value of TP99.9. • TP99.99: Remove the top 0.01% from the list, and the highest value left is the value of TP99.99.

Viewing a Real-Time Test Report

View the monitoring data of each metric during a pressure test through the real-time test report.

Prerequisites

The test task is being executed.

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **JMeter Projects** in the left navigation pane, and choose > **View Report** in the row containing the desired test project.

Step 2 On the **Performance Reports** tab page, click in the row containing the test plan whose real-time test report you want to view. For details about the parameters, see **Table 3-28**. Click **Stop Task** to stop the current task.

Step 3 On the **Overview** tab page, you can view the number of failed/total requests, average RT, maximum concurrency, success rate, bandwidth, dynamic trend, and response codes.

Step 4 On the **Detail** tab, you can view the logs, common test metrics, and request details of the test plan.

1. Click **View Log**. In the displayed dialog box, click Download Request Log to obtain the request log table. **Table 3-29** describes the table headings in the downloaded

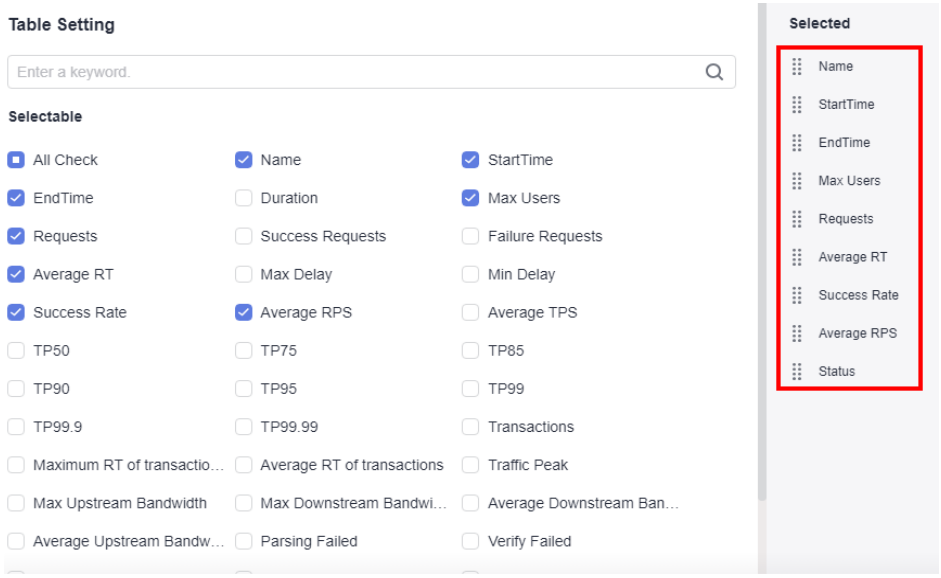
table. Ten request logs are displayed based on the name, return code, and result. There are no more than two request log files.

Table 3-29 Description of fields in the table header

Table Header Field	Description
Name	Name of a request
Result	Return result of the request. The value can be **FAILURE** or **SUCCESS**.
Begin Time	Start time of a request
End Time	End time of a request
URL	URL of a request
Method	Method for invoking a request
Request Header	Request header information
Request Body	Request body information
Response Code	Response code
Response Header	Response header information
Response Body	Response body information
Failure Message	Cause of a failed request

2. The sampling modes displayed in the report details are equidistant sampling value and equidistant average value.
   - Equidistant sampling value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with sampling points at a fixed interval.
   - Equidistant average value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with average values at a fixed interval.
3. Click the **Data View** drop-down list box and enter a keyword to search for the required thread group or request data. You can also click the thread group or request data to be displayed in the directory in the **Data View** drop-down list box.
4. On the **Detail** tab page, you can also click **List** to go to the report metric summary page.
   - Click **Customized Column**. In the displayed dialog box, select the items to be displayed, and drag the selected items in the **Selected** list on the right to change the item sequence.

Figure 3-14 Table settings



- Click  in the **Operation** column to view logs.
- Click  in the **Operation** column to edit a thread group.

----End

Viewing an Offline Test Report

After a pressure test is complete, the system generates an offline test result report.

Prerequisites

The test task is complete.

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **JMeter Projects** in the left navigation pane, and choose  > **View Report** in the row containing the desired test project.

Step 2 On the **Performance Reports** tab page, click  in the row containing the test plan whose test report you want to view. For details about the parameters, see **Table 3-28**. CodeArts PerfTest retains offline report data for three years. You can download an offline PDF report to a local PC and export the raw data (in CSV) for further processing.

Step 3 On the **Overview** tab page, you can view the number of failed/total requests, average RT, maximum concurrency, success rate, bandwidth, dynamic trend, and response codes.

Step 4 On the **Detail** tab, you can view the logs, common test metrics, and request details of the test plan.

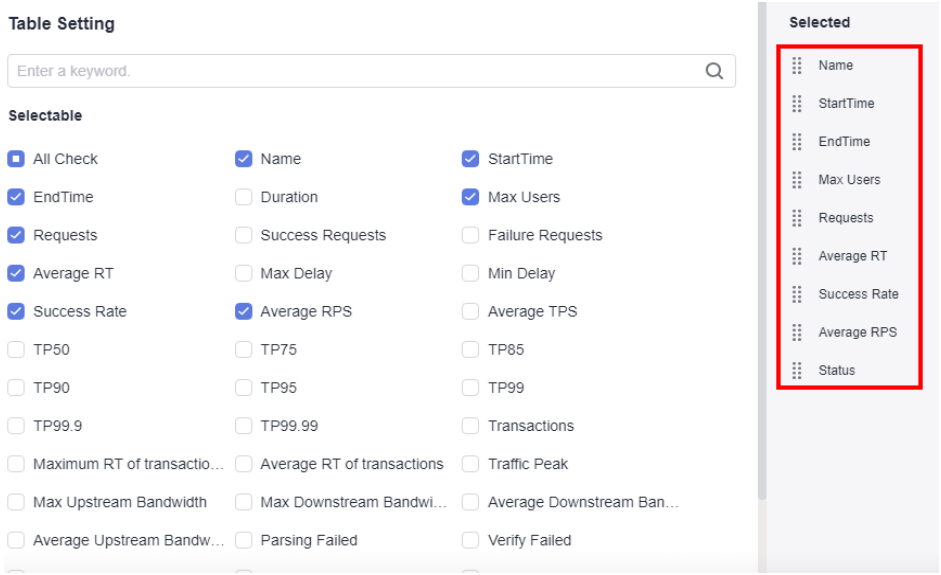
1. Click **View Log**. In the displayed dialog box, click Download Request Log to obtain the request log table. **Table 3-30** describes the table headings in the downloaded table. Ten request logs are displayed based on the name, return code, and result. There are no more than two request log files.

   Table 3-30 Description of fields in the table header

Table Header Field	Description
Name	Name of a request
Result	Return result of the request. The value can be **FAILURE** or **SUCCESS**.
Begin Time	Start time of a request
End Time	End time of a request
URL	URL of a request
Method	Method for invoking a request
Request Header	Request header information
Request Body	Request body information
Response Code	Response code
Response Header	Response header information
Response Body	Response body information
Failure Message	Cause of a failed request

2. The sampling modes displayed in the report details are equidistant sampling value and equidistant average value.
   - Equidistant sampling value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with sampling points at a fixed interval.
   - Equidistant average value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 45 minutes is displayed with average values at a fixed interval.
3. Click the **Data View** drop-down list box and enter a keyword to search for the required thread group or request data. You can also click the thread group or request data to be displayed in the directory in the **Data View** drop-down list box.
4. On the **Detail** tab page, you can also click **List** to go to the report metric summary page.
   - Click **Customized Column**. In the displayed dialog box, select the items to be displayed, and drag the selected items in the **Selected** list on the right to change the item sequence.

Figure 3-15 Table settings



- Click  in the **Operation** column to view logs.
- Click  in the **Operation** column to edit a thread group.

----End

Report Comparison

For the test reports generated at different time or under different conditions, you can view the test result by comparing the reports.

Procedure

Step 1 Log in to the CodeArts PerfTest console, choose **JMeter Projects** in the left navigation pane, and choose  > **View Report** in the row containing the desired test project.

Step 2 On the **Performance Reports** tab page, click the name of the desired plan or click  in the **Operation** column.

Step 3 On the **Comparison** tab page, select the desired test reports.

You can select a maximum of three offline reports for comparison. The first selected report is used as the baseline report.

Step 4 Select a thread group from the **Thread Group Comparison** box to compare the metrics of this thread group in different reports.

----End

3.10 Crontask

3.10.1 Creating a Crontask

Prerequisites

A PerfTest or JMeter project has been created, and test tasks are included.

Procedure

Step 1 Log in to the CodeArts PerfTest console. In the navigation pane, choose **Scheduled Test**. Then click **Create**.

Step 2 Set the basic information by referring to **Table 3-31**.

Table 3-31 Creating a crontask

Parameter	Description
Related Project	Select an existing PerfTest or JMeter project.
Related Task	Select the task for which you want to perform a scheduled pressure test. A task can be associated with only one scheduled pressure test task to be executed.
Crontask Name	Name of the crontask
Description	Description of the crontask
Resource Group	Select a test resource group for the crontask. • You can select a shared resource group or an existing private resource group for a PerfTest project. Shared resource groups of a user can be bound to only one pressure test task. • Only existing private resource groups can be selected for a JMeter test project.

Parameter	Description
Run Frequency	Frequency of running the crontask. Option: **Run only once** and **Run regularly**. • **Run only once**: The task is executed only once at the time set in **Date of running**. • **Run regularly**: The period settings can be **Monthly**, **Weekly**, and **Advanced**. – **Monthly**: The task is executed every month on the specified running date (one day or multiple days) and time. – **Weekly**: The task is executed every week on the specified running date (one day or multiple days) and time. – **Advanced**: The task is repeatedly executed based on crontab expressions. Use this mode if you are familiar with crontab expressions. The rules for entering running dates are as follows: Minute: integers ranging from 0 to 59, and operators * , - / Hour: integers ranging from 0 to 23, and operators * , - / Day (days in a month): integers ranging from 1 to 31, and the operator ? * , - / Month: Only ***** (monthly) is supported. Week (days in a week): any days in a week and the question mark (?) operator The day and week rules are mutually exclusive. When a non-default value is set for one parameter, the default value **?** is automatically set for the other parameter.
Deadline	Set this parameter only when **Run Frequency** is set to **Run regularly**. The task will not be executed after the deadline. The furthest deadline is one year later.

Step 3 When the configuration is complete, click **Sure**.

----End

3.10.2 Managing a Crontask

After a crontask is created, you can view, stop, and delete it.

Viewing a Crontask

You can view created crontasks.

Step 1 Log in to the CodeArts PerfTest console and choose **Scheduled Test** in the left navigation pane.

Step 2 You can view the overview of crontasks in the calendar on the left of the page. By default, crontasks for the day are displayed on the right of the page. Switch to the

other tab to view all crontasks. They can be sorted by the next running time and filtered by the name or status.

----End

Stopping a Crontask

After a crontask is started at the configured running time, you can stop the test if the pressure test plan changes.

Step 1 Log in to the CodeArts PerfTest console and choose **Scheduled Test** in the left navigation pane.

Step 2 On the **Crontask for the day** tab page in the right pane, click the crontask name to be stopped.

Step 3 On the **Crontask Detail** page, click  in the upper right corner.

Step 4 In the dialog box that is displayed, click **OK** to stop the crontask.

----End

Deleting a Crontask

Step 1 Log in to the CodeArts PerfTest console and choose **Scheduled Test** in the left navigation pane.

Step 2 On the **Crontask for the day** or **All crontask** tab page in the right pane, find the crontask to be deleted.

Step 3 Click  in the **Operation** column of the crontask to be deleted. In the displayed dialog box, click **OK** to delete it.

----End

3.11 Configuring SLAs

A Service Level Agreement (SLA) defines criteria for performance metrics. You can gain a better understanding of the performance of your service by checking if the metrics meet the criteria specified in the SLA.

Only PerfTest projects support SLAs. JMeter projects do not support. When the SLA metric thresholds are met, alarms will be generated or the performance tests will be stopped.

During a CodeArts PerfTest pressure test, the SLA sampling interval is 6 seconds. Any request that does not receive a response within this interval is not accounted for in the SLA metrics. If numerous requests experience high latency during the stress test, it may impact the SLA assessment.

You can create SLAs in either of the following ways:

- **Creating and Importing an SLA Template**. SLA templates can be imported for any PerfTest test tasks.

- **Create an SLA for a specific test case**. The SLA is applicable only to this test case.

Creating and Importing an SLA Template

Step 1 Log in to the CodeArts PerfTest console, choose **SLAs** in the navigation pane, and click **Create SLA Template**. You can create up to 20 SLA templates in your account in each region.

Step 2 Set the SLA basic information.

- **SLA Name**: This is used to identify an SLA. The SLA name must be unique in a region.
- **SLA Description**: This parameter is optional. The value contains a maximum of 255 characters.

Step 3 Click **Add Rule**. On the **Create Rule** dialog box that is displayed, set the SLA rule by referring to **Table 3-32** and click **Confirm**.

Table 3-32 SLA rule parameters

Parameter	Description
Rule Name	This is used to identify a rule.
Metric	Three types of metrics are supported. CodeArts PerfTest will generate an alarm or stop a performance test when the metric reaches the specified threshold. • Average response time (RT) • Requests per second (RPS) • Success rate
Thresholds	Thresholds for triggering an alarm or stopping a performance test. For example, if you set the metric threshold for **Stop Test** to **< 100** and **3 consecutive times**, the test task will be stopped when the metric exceeds 100 for three consecutive times.

Step 4 (Optional) To create multiple rules, repeat **Step 3**. Up to five rules can be created for each SLA.

Step 5 Click **Save** to complete the SLA template creation. You can view the created SLA template on the **SLAs** page. To edit or delete an SLA template, locate the row that contains the SLA template, and click **Edit** or **Delete**.

Step 6 In the navigation pane, choose **PerfTest Projects**.

Step 7 Click the name of the desired project to go to the details page.

Step 8 In the **Case List** on the left, select the case for which the SLA rule is to be configured.

Step 9 Click the **SLA Configuration** tab and click **Import from an existing SLA template**. In the **Import SLA Template** area, select the name of the created SLA template and import the SLA template.

Step 10 In the **SLA Rules** area, select one or more requests to which the SLA rules will be applied.

You can also edit, delete, or add rules as required.

Step 11 Click **Save** to finish the SLA configuration.

----End

Creating SLA Rules for a Test Case

Step 1 Log in to the CodeArts PerfTest console and choose **PerfTest Projects** in the left navigation pane.

Step 2 Click the name of the desired project to go to the details page.

Step 3 In the **Case List** on the left, select the case for which the SLA rule is to be configured.

Step 4 Click the **SLA Configuration** tab.

Step 5 Click **Create SLA**.

Step 6 Click **Add Rule**. On the **Create Rule** dialog box that is displayed, set the SLA rule by referring to **Table 3-33** and click **Confirm**.

Table 3-33 SLA rule parameters

Parameter	Description
Rule Name	This is used to identify a rule.
Metric	Three types of metrics are supported. CodeArts PerfTest will generate an alarm or stop a performance test when the metric reaches the specified threshold. • Average response time (RT) • Requests per second (RPS) • Success rate
Thresholds	Thresholds for triggering an alarm or stopping a performance test. For example, if you set the metric threshold for **Stop Test** to **< 100** and **3 consecutive times**, the test task will be stopped when the metric exceeds 100 for three consecutive times.

Step 7 In the **Target Requests** column, select one or more requests for each rule.

Step 8 (Optional) To create multiple rules, repeat **6** to **7**. Up to five rules can be created for each SLA.

Step 9 Click **Save** to finish the SLA configuration.

----End

3.12 Reference

3.12.1 Header Description

Table 3-34 Header Description

Protocol Header Name	Description
Accept	Acceptable types of response contents.
Accept-Charset	Acceptable character sets.
Accept-Datetime	Acceptable versions displayed by time.
Accept-Encoding	Acceptable encoding methods.
Accept-Language	Acceptable natural languages of response contents.
Authorization	Information used for HTTP authentication.
Cache-Control	Instructions that must be followed by all cache mechanisms for a request or response chain.
Connection	Priority connection types for a browser.
Content-Length	The length of the request body represented by 8-byte arrays.
Content-MD5	The binary MD5 hash value of the content of the request body, which is encoded using Base64.
Content-Type	Multimedia types of the request body (used in POST and PUT requests).
Cookie	An HTTP cookie sent by servers through **Set-Cookie**.
Date	The date and time when a message is sent.
DNT	A request for a web application to stop tracking a user. In the Firefox browser, it is equivalent to the X-Do-Not-Track protocol header field (supported for Firefox/4.0 Beta 11 or later). Safari and Internet Explorer 9 also support this field.
Expect	Specific actions required by a client for a server.
Front-End-Https	Refers to non-standard header fields used by Microsoft servers and load balancers.
From	The email address of the user who initiates the request.

Protocol Header Name	Description
Host	Domain name of the server (used for virtual host), and port number of the transmission control protocol listened to by the server. If the requested port is the standard port of the corresponding service, the port number can be omitted. This field is mandatory since HTTP/1.1. If the domain name in the URL is an IP address, this field is automatically added. Otherwise, enter the IP address and port number of the tested application in this field.
If-Match	The corresponding operation is performed only when the entity provided by the client matches the entity on the server. It is used in a method such as **PUT** to update a resource which has not been modified since the last update.
If-Modified-Since	Returning of **304 Not Modified** is allowed when the corresponding content is not modified.
If-None-Match	Returning of **304 Not Modified** is allowed when the corresponding content is not modified. Refer to the HTTP entity tag. In a typical use, when a URL is requested, the web server returns the resource and its corresponding **ETag** value, which is placed in the **ETag** field of the HTTP. The client can then decide whether to cache the resource and its **ETag**. If the client wants to request the same URL again, it sends a request that contains the saved **ETag** and the **If-None-Match** field.
If-Range	If an entity is not modified, send one or more parts that are missing to the sender. Otherwise, send the entire new entity.
If-Unmodified-Since	A response is sent only when the entity has not been modified since a specific time.
Max-Forwards	Refers to the number of times a message can be forwarded by the proxy and gateway.
Origin	Refers to a sharing request initiated for resources of different origins. The server is required to add an **Access-Control-Allow-Origin** field to the response.
Pragma	Related to specific implementations and may produce multiple effects at any time in the request or response chain.
Proxy-Authorization	Information used to authenticate a proxy.
Proxy-Connection	Derives from the errors in the implementation of an early HTTP version. The function of this field is the same as that of the standard Connection field.
Proxy-Password	Proxy password.

Protocol Header Name	Description
Proxy-Server	Proxy service.
Proxy-Username	Proxy username.
Range	Requests only a part of an entity with the byte offset starting from 0.
Referer	Refers to the previous page accessed by a browser. A link on this page brings the browser to the currently requested page.
TE	Refers to the transmission coding mode expected by a browser. You can use a value of **Transfer-Encoding** in the response protocol header. In addition, the value **trailers** (related to the block transmission mode) indicates that the browser expects to receive additional fields if the size of the last block is 0.
Upgrade	The server needs to be upgraded to another protocol.
User-Agent	Refers to the character string of the browser identity.
Via	Refers to request-sending agents informed to a server.
Warning	Refers to a general warning indicating that errors may exist in the body of an entity.
X-Wap-Profile	Refers to an XML file linked to the Internet. The file describes devices being connected.
X-Requested-With	Used to identify Ajax and XML requests. Most JavaScript frameworks send this field and set its value to **XMLHttpRequest**.
X-Http-Method-Override	A web application is requested to use the method (usually PUT or DELETE) specified in the protocol header field to override the method specified in the request (usually POST). Use this method when a browser or firewall prevents direct sending of the PUT or DELETE method (may be caused by a vulnerability in the software, which needs to be fixed, or because a configuration option is required, in which case the situation should not be bypassed).
X-Forwarded-Proto	Refers to a fact standard used to identify the protocol used at the beginning of an HTTP request.
X-Forwarded-Host	Refers to a fact standard used to identify the host request header originally sent by the client.
X-Forwarded-For	Refers to a fact standard used to identify the original Internet address of a client that is connected to a web server through an HTTP agent or load balancer.

Protocol Header Name	Description
X-Csrf-Token	The **X-CSRFToken** or **X-XSRF-TOKEN** header is used to prevent cross-site request forgery.
X-ATT-DeviceId	Enables the server to easily interpret the common device models and firmware information in the **User-Agent** field of the AT&T device.

3.12.2 Regular Expression Metacharacters

Table 3-35 Metacharacter description

Metacharacter	Description
.	Matches any characters except **\n**. If **\n** needs to be included, use other modes such as **[\s\S]**.
^	Matches the start position of an input character string and does not match any characters. Use **\^** to match the character itself.
$	Matches the end position of an input character string and does not match any characters. Use **\$** to match the character itself.
*	The preceding characters or sub-expressions are matched zero or more times. ***** is equivalent to {0,}. For example, **\^*b** can match **b**, **^b**, **^^b**, and so on.
+	Matches preceding characters or sub-expressions one or more times, equivalent to **{1,}**. For example, **a+b** can match **ab**, **aab**, **abb**, **aaab**, and so on.
?	Matches preceding characters or sub-expression zero times or once, equivalent to **{0,1}**. For example, **a[cd]?** can match **a**, **ac**, and **ad**. When this character follows any other qualifier such as *****, **+**, **?**, **{n}**, **{n,}**, or **{n,m}**, the matching mode is **non-greedy**. **Non-greedy** mode matches the shortest possible searched character strings, and the default **greedy** mode matches the longest possible searched character strings. For example, the character string **oooo**, **o+?** matches only a single **o**, while **o+** matches all **o**.
|	The logic **or** is performed on two matching conditions. For example, the regular expression **(him|her)** matches **it belongs to him** and **it belongs to her**, but cannot match **it belongs to them**.
\	Marks the next character as a special character, text, reverse reference, or octal escape character. For example, **n** matches the character **n**, **\n** matches the newline character, **** matches ****, and **\(** matches **(**.

Metacharacter	Description
\w	Matches a letter, digit, or underscore (_).
\W	Matches any character that is not a letter, digit, or underscore (_).
\s	Matches any blank character, such as a space, tab character, or form feed. It is equivalent to **[\f\n\r\t\v]**.
\S	Matches any character except blank characters and is equivalent to **[^\f\n\r\t\v]**.
\d	Matches any digit and is equivalent to **[0–9]**.
\D	Matches any non-digital character and is equivalent to **[^0-9]**.
\b	Matches a word boundary (the position between a word and a space) and does not match any characters. For example, **er\b** matches **er** in **never** but does not match **er** in **verb**.
\B	A non-word boundary match. For example, **er\B** matches **er** in **verb** but does not match **er** in **never**.
\f	Matches a form feed and is equivalent to **\x0c** and **\cL**.
\n	Matches a linefeed and is equivalent to **\x0a** and **\cJ**.
\r	Matches a carriage return character and is equivalent to **\x0d** and **\cM**.
\t	Matches a tab character and is equivalent to **\x09** and **\cI**.
\v	Matches a vertical tab character and is equivalent to **\x0b** and **\cK**.
\cx	Matches control characters indicated by **x**. For example, if **\cM** matches **Control-M** or a carriage return character, the value of **x** must be between A–Z or a–z. Otherwise, the **c** character indicates **c** itself.
{n}	The value **n** is a non-negative integer and refers to the number of matching times. For example, **o{2}** does not match **o** in **Bob**, but matches **o** in **food**.
{n,}	The value **n** is a non-negative integer and refers to the minimum number of matching times. For example, **o{2,}** does not match **o** in **Bob** but matches all **o** in **fooooood**. **o{1,}** is equivalent to **o+**, and **o{0,}** is equivalent to **o***.
{n,m}	The values **n** and **m** (n≤m) are non-negative integers, where **n** refers to the minimum number of matching times and **m** refers to the maximum matching times. For example, **o{1,3}** matches the first three **o**s in **fooooood**, and **o{0,1}** is equivalent to **o?**. Note that a space cannot be inserted between commas and digits. For example, **ba{1,3}** can match **ba**, **baa**, or **baaa**.

Metacharacter	Description
x|y	Matches **x** or **y**. For example, **z|food** matches **z** or **food**, and **(z|f)ood** matches **zood** or **food**.
[xyz]	Refers to a character set that matches any characters included. For example, **[abc]** matches **a** in **plain**.
[^xyz]	Refers to a reverse character set that matches any characters except **xyz**. For example, **[^abc]** matches **p** in **plain**.
[a-z]	Refers to a character range and matches any characters in the specified range. For example, **[a-z]** matches any lowercase letters from **a** to **z**.
[^a-z]	Refers to a reverse character range and matches any characters not in the specified range. For example, **[^a-z]** does not match any characters from **a** to **z**.
( )	Defines expressions between **(** and **)** as **group** and save characters that match the expression to a temporary area. A maximum of nine characters can be saved in a regular expression, and these characters can be referenced by symbols **\1** to **\9**.
(pattern)	Matches pattern and captures sub-expressions of the match. You can use the $0–$9 attribute to retrieve captured matches from the result **match** set.
(?:pattern)	Matches pattern but does not capture sub-expressions of the match. That is, it is a non-capture match and does not store matches for future use. This is useful for the **or** character combined with **(|)**. For example, **industr(?:y|ies)** is a simpler expression than **industry|industries**.
(?=pattern)	Refers to a non-capture match and indicates a forward positive pre-check, searching character strings at the start position of any character strings that match pattern. There is no need to capture the match for future use. For example, "Windows(?=95|98|NT|2000)" matches "Windows" in "Windows2000", but does not match "Windows" in "Windows3.1". A pre-check does not consume characters. That is, after a match occurs, the next search starts immediately, instead of starting from pre-checked characters.
(?!pattern)	Refers to a non-capture match and indicates a forward negative pre-check, searching character strings at the start position of any character strings that do not match pattern. There is no need to capture the match for future use. For example, "Windows(?=95|98|NT|2000)" matches "Windows" in "Windows3.1", but does not match "Windows" in "Windows2000".

To match special characters, add \ before the special characters. For example, to match the following special characters: ^, **$**, **()**, **[]**, **{}**, **.**, **?**, **+**, *, and |, use **\^**, **\$**, **\ (**, **\)**, **\ [**, **\]**, **\{**, **\}**, **\.**, **\?**, **\+**, *****, and **\|**.

3.12.3 Modifying an Exported Project File

Table 3-36 Request composition of think time

Parameter	Description
name	Can be modified.
t	*Duration (ms) **Think Time** refers to the waiting time between two consecutive operations performed by a user, such as the interval between login and search. Suppose that the response time for each running of a transaction is 0.5s. ● To execute two transaction requests per second, do not set the **Think Time** request composition. ● To execute only one transaction request per second, set **Think Time** to **1s**. If the think time is set to 1s and the response time is longer than 1s, the think time will not take effect, and pressure test requests are sent according to the response time.

Table 3-37 Request composition of packet

Parameter	Description
name	Can be modified.
http_version	Protocol type. HTTP, HTTPS, TCP, and UDP protocols are supported.
When **Protocol Type** is **HTTP** or **HTTPS**, set the following parameters.	
method	GET, POST, PATCH, PUT, and DELETE are supported.
return_timeout	Timeout period for waiting for a response from the server when a request is sent. If this parameter is not set, the default response timeout interval (5,000 ms) is used.
URL	URL for sending a request, for example, **http://***domain name***/***path*, or **http://***domain name***/path?key1=value1&key2=value2**.
headers	Headers consist of keywords and values and are used to notify servers of clients' requests. For details about headers, see **Header Description**.

Parameter	Description
Packet content	The body of an entity contains a data block consisting of random data. Not all packets contain the body of an entity. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values. When the request mode is **GET**, packet content cannot be input.
When **Protocol Type** is **TCP**, set the following parameters. TCP packets do not support the response extraction and checkpoint functions.	
IP	IP address of the tested server to which requests are sent.
port	Port number of the tested server to which requests are sent.
connect_timeout	Timeout duration for the server's response after a connection is initiated.
return_timeout	Timeout duration for waiting for the server's response after a connection is established.
Connection Settings	• **Repeated use**: When a request response is complete, the connection remains and is used to send and receive the next request response. • **Close**: When a request response is complete, the connection is closed and re-established next time.
check_end_type	Used to judge whether a request response has been received. It is recommended that a unique end-of-text character be set. If multiple such characters are present in a response, the response is considered complete when the first character is received. As a result, the received response data may be incomplete. • **Length of returned data**: length of the returned data, in bytes. When a response of this length is received, data receiving is complete. • **End-of-text character**: ending mark of the returned data. When an end-of-text character is received, data receiving is complete.

Parameter	Description
body	The body of an entity contains a data block consisting of random data. Not all packets contain the body of an entity. Content format: Select character strings or a hexadecimal code stream based on the service request content of the tested server. A hexadecimal code stream can contain only numbers 0–9 and letters a–f. The total number of characters must be an even number. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values.

Table 3-38 Request composition of response extraction

Parameter	Description
name	Must be unique. The value of the response extraction is assigned to this variable.
range	Range of the content to be extracted. • Packet content • Header • URL • Response Code The packet content, header, and URL can be extracted using regular expressions.
regexp	A regular expression specifies a type of logical expressions performed on strings. In a regular expression, you use certain predefined strings and a combination of these strings to form a rule string that is used to specify a specific filtering logic. A complete regular expression consists of two types of characters: special characters (also called meta characters) and literal or common text characters (such as letters, digits, and underscores). For details about meta characters, see **Regular Expression Metacharacters**. Parentheses **()** indicate extraction and are used to enclose content to be extracted. The content in a pair of parentheses forms a sub-expression.
match_index	This parameter indicates the sequence number of the matched content extracted through a regular expression. This parameter cannot be set to **0**. Value: a positive integer

Parameter	Description
exp_index	This parameter indicates the sequence number of the parsed sub-expression. • Value **0** indicates that the entire regular expression is matched. • Value **1** indicates that the first sub-expression of the regular expression is matched, that is, the content extracted by the first **()**. Value range: natural numbers After extracting content through **Regular Expression** and **Sequence Number of Matching Item**, use **Expression Value** to obtain the final content.
JSON Key Name	Enter the key name to be obtained. For example, {key:{"key1":"v1","key2":["v2","v3"]}}. If you want to obtain **v2**, enter **key.key2[0]**.
default	Indicates the value returned when a regular expression match fails.

Table 3-39 Request composition of checkpoint

Parameter	Description
name	Can be modified.
value	This parameter is used to respond to HTTP, HTTPS, TCP and UDP response status codes carried in a packet, including 1XX, 2XX, 3XX, 4XX, and 5XX.
header_checks	Headers of the HTTP, HTTPS, TCP, and UDP.
body_checks	Body parts of HTTP, HTTPS, TCP, MQTT, and UDP, namely the load parts of the requests and responses.

Table 3-40 Adding a test project

Parameter	Description
name	Indicates the test project name.
description	Indicates description of the test project.

Table 3-41 Adding a transaction

Parameter	Description
name	Indicates the transaction name.
contents	Request content. You can add transaction requests under a transaction based on service requirements.

Table 3-42 Adding a task

Parameter	Description
issue_num	Number of concurrent users. The number of concurrent users refers to the number of users performing operations on the system at the same time. In CodeArts PerfTest, it is the number of virtual users set when you define test task phases.
name	Phase name. Set a name for the service scenario, for example, **homepage test**.
time	Duration (second). The maximum time for performing a pressure test in the current phase.
count	Total number of transmissions. During the running of a task, the transaction is calculated based on the number of running times. When the specified value is met, the performance test of the transaction under the task is terminated.

Table 3-43 Adding a global variable

Parameter	Description
name	Global variable name.
variable	Global variable value. Variable content is transmitted in plain text during the pressure test. Exercise caution when entering the content.

Parameter	Description
variable_type	Global variable type. When the variable type is **Integer**, enter the value range of the variable. A pressure test task reads the value range of the corresponding global variable when it is running. For each virtual user, the variable value is polled. That is, the first virtual user obtains the first value of the variable, and the second virtual user obtains the second value. If the values are exhausted, the next virtual user obtains the first value again. Add multiple variables as required. A pressure test task reads the value range of the corresponding global variable when it is running. For each virtual user, the variable value is polled. That is, the first virtual user obtains the first value of the variable, and the second virtual user obtains the second value. If the values are exhausted, the next virtual user obtains the first value again.

3.12.4 Mapping Between JMeter and PerfTest Fields

CodeArts PerfTest supports the import and automatic switchover of the JMeter script. **Table 3-44** describes the mapping between JMeter and PerfTest fields.

Table 3-44 Mapping between JMeter and PerfTest fields

JMeter Field		PerfTest Field	Parameter Description
Thread group	Name	Transaction Name/ Task Name	The HTTP request and the subsequent HTTP request form a transaction. If no valid value is obtained, number of concurrent requests is 1 and the duration is 1 minute by default.
	Number of threads	Number of concurrent test tasks	
	Duration	Test task duration	
User-defined variables	Name	Enumeration variable name	Global variable, enumerated type. The name must be unique.
	Value	Enumeration variable value	

JMeter Field		PerfTest Field	Parameter Description
User parameters	Name	Enumeration variable name	Global variable, enumerated type. The name must be unique.
	User_1	Enumeration variable value	
	User_2	Enumeration variable value	
	...	Enumeration variable value	
	User_N	Enumeration variable value	
HTTP header manager	Name	Packet header	When the POST, PATCH, PUT, or DELETE request mode does not contain the **Content-Type** header, the **Content-Type** header value is set to **application/x-www-form-urlencoded** by default.
	Value	Header value	
Default HTTP request	Protocol	-	If the HTTP request does not contain a value, the default value of the HTTP request is used. Priority: HTTP request > thread group > test plan.
	Server name or IP address		
	Port number		
	Path		
	Parameter		
	Message body data		
	Response timeout in the advanced options.		

JMeter Field		PerfTest Field	Parameter Description
HTTP request	Protocol	Protocol type	Packet. Ignore request modes except POST, GET, PATCH, PUT, and DELETE. The default value is 5,000 ms when no response timeout value is obtained.
	Protocol, server name or IP address, port number, and path	Request URL	
	Method	Request mode	
	Parameter	Add the content to the URL or packet content as required.	
	Message body data	Packet content	
	Response timeout in the advanced options.	Response timeout	
Regular expression extractor (the response field to be checked)	Body	Packet content	Response extraction. Only the regular expression extractor and fixed timer under the HTTP requests can be imported. If the default value is empty, the reference name is used by default.
	Body (unescaped)	Packet content	
	Body as a Document	Packet content	
	Message header	Header	
	URL	URL	
	Response code	Response code	
	Request Headers. Response information.	Import is not supported. The regular expression extractor is ignored.	
Regular expression extractor	Reference name	Variable name	
	Regular expression	Regular expression	
	Template	Sequence number of matching item	
	Match digits	Expression value	
	Default value	Default value	
Fixed timer	Thread latency	Duration	Think time. If no valid value is read, the default value 1,000 is used.

JMeter Field		PerfTest Field	Parameter Description
Random function	-	Random number in a range	An integer generated randomly within the range entered by a user. For example, ${__Random(-2147483648,2147483647)}
JSON extractor	Name of created variables	Variable name	Response extraction. Default extraction range: parameter values in .json format. Only the JSON extractor under the HTTP requests can be imported.
	JSON Path expressions	Key name	
	Default Values	Default value	

4 FAQs

4.1 Resource Group Management

4.1.1 Suggestions on Test Resource Configuration

Test Resource Groups and Their Constraints

- Test resource groups are classified into shared resource groups and private resource groups. Shared resource groups are provided by the system by default, and private resource groups need to be created.
- Execution nodes of the shared resource group have been bound with an elastic IP address (EIP). When the tested application has network access restrictions, use a private resource group.
- In the same region, a shared resource group supports PerfTest test tasks with up to 10,000 or 100,000 concurrent users in total (determined by the package you selected when installing CodeArts PerfTest). If the number of concurrent users exceeds the upper limit, use a private resource group.
- JMeter test tasks can use only private resource groups.

Suggestions on Using Nodes

- If an application is deployed on a node in a cluster, the node cannot be selected to create a private resource group. Do not run any applications or perform other functions on nodes used for test resource groups. Otherwise, applications may run abnormally.
- If you want to perform pressure tests on external services, bind an EIP to each execution node. If you want to debug external services, bind EIPs to both the debugging node and execution node. The test bandwidth is limited by the EIPs' bandwidth.
- Create at least three empty nodes. Two are for debugging an execution node. The other is the execution node/executor (a target machine that a pressure test will be performed on and can provide performance data during testing). Create nodes of the required specifications based on the number of concurrent users for a pressure test. For details about the recommended node

specifications, see **Table 4-1** and **Table 4-2**. These specifications are for reference only. Resource specification requirements for a pressure test are influenced by think time, protocol type, the size and number of requests and responses, response time, and result verification. Adjust the specifications as needed.

- In a PerfTest test project, one execution node with 8 vCPUs and 16 GB memory supports 10,000 concurrent users. In a JMeter test project, one execution node with 8 vCPUs and 16 GB memory supports 2,000 concurrent users.

Table 4-1 Recommended node specifications for PerfTest projects

Concurrent Users	Specifications	Quantity
0–5,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 4 vCPUs | 8 GB	1
5,001–10,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	1
10,001–20,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	2
20,001–30,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	3
30,001–40,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	4
40,001–50,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	5
More than 50,001	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	*n*

Table 4-2 Recommended node specifications for JMeter projects

Concurrent Users	Specifications	Quantity
0–1,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 4 vCPUs | 8 GB	1
1,001–2,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	1
2,001–4,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	2
4,001–6,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	3
6,001–8,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	4
8,001–10,000	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	5
More than 10,001	Debugging node: 4 vCPUs | 8 GB	2
	Execution node: 8 vCPUs | 16 GB	*n*

4.2 Pressure Test Project Management

4.2.1 What Are the Differences Between Think Time and Duration in CodeArts PerfTest?

There are two time-related concepts in CodeArts PerfTest:

- Think time (ms): waiting period between two consecutive operations performed by a user.

- Duration (min): time spent executing a test task.

The think time does not affect the total duration of concurrency. It only affects the number of concurrent requests. The following example explains this in detail.

The think time is set to 1,000 ms, the duration is set to 10 min, and the number of concurrent users is set to 10. The formula for calculating the number of concurrent requests is: Duration x Number of concurrent users/Think time. The result is 6,000 (600s x 10/1s). The total duration of the concurrency for this task is 10 min, and 6,000 query requests are sent to the server. In some cases, the test result shows that the number of concurrent requests is smaller than 6,000. The reason is that if a message is not immediately responded to, the system waits 0.1s for response.

4.2.2 What Is the Number of Concurrent Users?

A pressure test simulates actual service operations of users, while the number of concurrent users refers to the number of users who perform service operations on the system simultaneously.

For example, when a game website holds a competition at a certain time, devices are expected to support a large number of concurrent users. In this case, the number of concurrent users can be set to simulate the number of users performing operations at the same time.

The number of concurrent users, the concurrency duration, and think time are used to calculate the number of concurrent requests and the maximum concurrent requests supported by the server. These numbers are compared with the desired numbers to determine whether the customer requirements can be met.

4.2.3 How Do I Fill in Packets?

Packets refer to all click operations on the website. A click operation is edited to a code stream complying with protocol specifications and carrying a user's request before the code stream is sent to a third party, leading to a correct or failed response. A correct response indicates that the operation is successful, and a failed response provides tips for rectifying the problem.

CodeArts PerfTest supports packets in the following request types: GET, POST, PATCH, PUT, and DELETE. The following describes how to fill in packets.

1. Before a pressure test, confirm the request type of an operation.

   Taking queries as an example, query messages are GET requests. You can set the request mode to GET during configuration.

2. What do I do if my request messages require parameter input?

   If a request involves various fields, press **F12** or use a packet-capturing tool (such as Wireshark) to check how a packet is requested, what the body format is, and how the request is transmitted to third-party application programming interfaces (APIs). Then, fill in the packet to be tested according to the actual service.

   Generally, such a request uses the POST method. After this method is selected, the following information is displayed.

Figure 4-1 Packet content

For the standard HTTP or HTTPS format, fill in the packet header based on the captured content. The packet body specifies the request content, which depends on the service to be tested. The body can be a game login request or a registration request. All packets can be edited for pressure tests as long as the request complies with the HTTP or HTTPS protocol.

The preceding example is also applicable to PATCH, PUT, and DELETE methods. First, confirm the protocol type, request method, and request link of the application to be tested, and then confirm the content in a request.

4.2.4 Why Does Transaction Debugging Frequently Fail?

Ensure that the following conditions are met before debugging:

- The resource group is running.
- The network between the debugging node of the resource group and the tested application is normal.
  a. Log in to the Elastic Cloud Server (ECS) management console.
  b. Find and log in to the debugging and execution nodes.
  c. Run the **curl *url*** command (*url* is the URL of the tested application) to check whether the network is normal.

If the preceding conditions are met, debug the transaction. Click **View Log** to check whether the returned content is correct.

If the returned content contains error information, check that the entered parameters and configured contents of the packet are correct.

4.2.5 Which Headers Are Mandatory in an HTTP-based Packet Request?

CodeArts PerfTest does not have mandatory headers. It only transparently transmits your defined headers.

The headers that must be carried in an HTTP request depend on whether the tested server verifies or uses these headers.

Therefore, add headers and bodies accordingly.

4.2.6 Why Does the CPU Usage of the Execution Node Used for the Pressure Test Remain High?

CodeArts PerfTest requires low processing latency.

The server may have a short response time for the sent packets, which requires continuous polling to reduce latency deviation, thus keeping CPU usage high.

Nodes used for providing pressure tests to resource groups are exclusive. Therefore, the high CPU usage does not affect your application or affect the test.

4.2.7 What Are the Differences Between Global Variables and Variables Extracted from Responses?

They are applicable to different scopes:

- Global variables can be used for a test project. Currently, enumeration, integer, and file variables are supported.
- Variables extracted from responses are local variables, which can be used only in the current transaction or in the current test case. The variables can be used in subsequent requests, but cannot be used across transactions or test cases.

4.2.8 What Is the Impact of the Bandwidth Applied for CodeArts PerfTest on Tests?

The required bandwidth depends on the request and response models of a pressure test.

For example, if the transactions per second (TPS) is 5,000 and each request packet is 1 KB, the total uplink bandwidth required is 5,000 KB. You can use the same method to estimate the downlink bandwidth.

The bandwidth limit only restricts the uplink bandwidth. Therefore, requests with bodies, such as POST and PUT, will consume more bandwidth resources.

If the bandwidth is insufficient during the pressure test, network packet loss occurs. The test report shows a higher latency and occurrence of timeout.

4.2.9 What Are the Differences Between a JMeter Test Project and a PerfTest Project?

The scripts that can be imported to JMeter and PerfTest projects are different.

- You can directly import JMeter scripts to a JMeter project and use the native JMeter engine to initiate a performance test.
- A PerfTest project supports the import of PerfTest scripts. You can also edit test tasks based on the actual performance test scenario.

4.2.10 How Do I Check If the Global Variable Values Are Read Sequentially in a Test Task?

Perform the following steps:

1. Create a global variable. It is recommended that the number of its values be no more than 10 (for example, 6, 5, 4, 3, 2, and 1) for quick testing.
2. Create a case, reference the global variable configured in **1** in the body of the case packet, set **Execution Policy** to **Count**, set the number of concurrent users to 1, and set the number of transmissions to 10. Then start the test task.
3. On the **Detail** tab page of the performance report, click **Log View**. In the displayed dialog box, click 

to download logs. Check whether the values of global variables in the request body in the request log are in the configured sequence.

   When reading proceeds to the last value, it will start from the first value again.

4.3 Pressure Test Report Management

4.3.1 What Are the Differences Between RPS and TPS in a CodeArts PerfTest Report?

RPS is short for requests per second. RPS = Total requests of a case/Running duration of the case.

TPS is short for transactions per second. TPS = Number of transactions of a case/Running duration of the case. During a CodeArts PerfTest pressure test, all test steps in a case are executed cyclically. Each cycle is regarded as a transaction.

In CodeArts PerfTest, the TPS is calculated based on the average number of request packets that are responded to per second in each statistical period (10s). For example, if 1,000 requests are responded to within 10 seconds, the TPS is 100. Some test tools collect only the number of requests sent per second, which cannot accurately reflect a system's capability. The TPS in CodeArts PerfTest collects the number of requests processed and returned by the system.

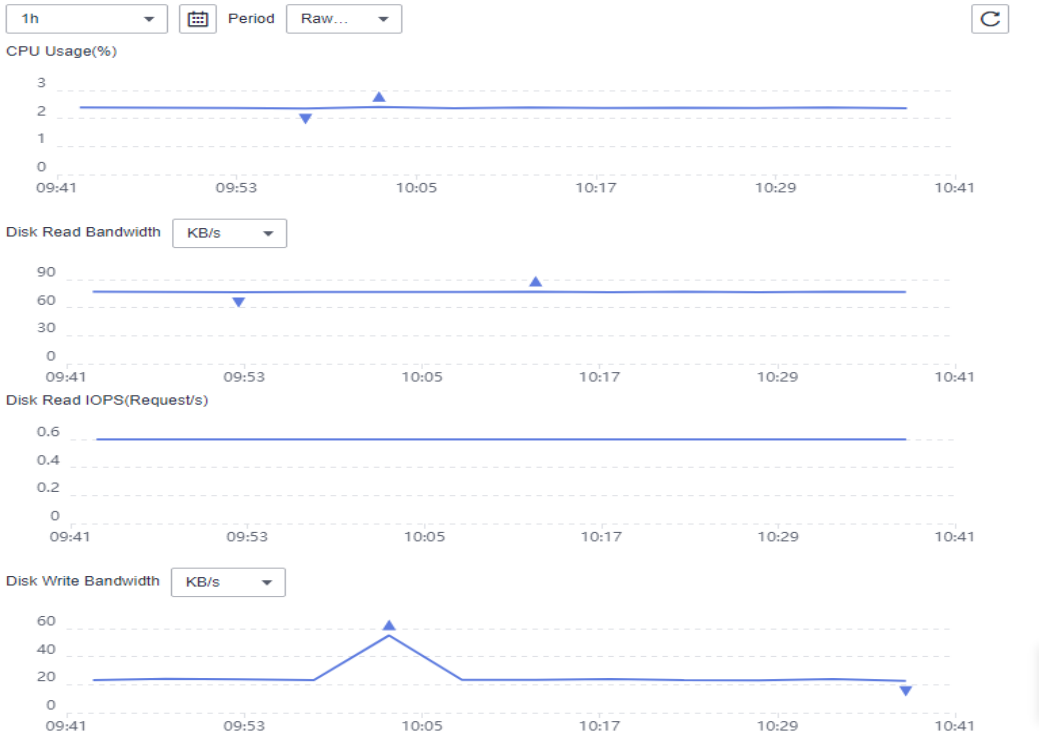
4.3.2 What Are the Meanings of Log Errors in a CodeArts PerfTest Report?

dialing to the given TCP address timed out

If the read/write request times out, check whether the network between the client and the server is reachable. If a shared resource group is used, check whether the network of the tested server is accessible from public networks. If a private resource group is used, check whether the network between the executor and the tested server is connected.

look up XXX timeout

If the network is abnormal, check the network load of the executor and server. You can check the network bandwidth usage of other cloud services, such as EIP.



not look like a TLS handshake

The server uses the HTTP protocol. Check whether the HTTPS protocol is mistakenly used for the test case.

4.4 General FAQs

4.4.1 Does CodeArts PerfTest Support Windows Server 2016 Standard (64-bit)?

CodeArts PerfTest and tested services support the following OSs:

- Currently, CodeArts PerfTest can be deployed only in Linux OSs, not in Windows OSs.
- Pressure tests can be performed in services as long as the network is normal.

4.5 Using JMeter Projects

4.5.1 What Are the Differences Between the JMeter Engine of CodeArts PerfTest and the Open-source JMeter?

The JMeter engine of CodeArts PerfTest is based on the open-source Apache JMeter. The default version is 5.4. You can also upload versions 5.2 and 5.3.

Compared with the local open-source JMeter, the JMeter engine of CodeArts PerfTest has the following advantages:

1. Automated distributed scheduling
2. Aggregated and visualized test results
3. Distributed multi-phase capability

4.5.2 What Scripts Does the JMeter Engine of CodeArts PerfTest Support?

The JMeter engine supports the following scripts:

- JMX scripts that are created by JMeter 5.2 to 5.4 and do not use any third-party plug-ins.
- Most scripts that use third-party plug-ins can be uploaded as JAR packages and that are implemented without modifying the ThreadGroup. However, CodeArts PerfTest does not guarantee the proper functioning of these scripts. You will need to debug them in CodeArts PerfTest.

4.5.3 Which Operations in Scripts Are Not Supported by the JMeter Engine of CodeArts PerfTest?

- Log output (Only requesting logs is supported.)
- Variables on the thread group configuration page

4.5.4 What Are the Possible Causes of a JMX File Import Error in a JMeter Test Project?

The possible causes of the error are as follows:

- The JMX file contains garbled characters.
- The JMX file contains third-party plug-ins and cannot be imported.

4.5.5 What Are the Suggestions for Using CodeArts PerfTest Scripts?

- Result viewers are not recommended in scripts.

  Different result viewers have different impacts on the pressure test performance. If you want to use result viewers, evaluate the risks.
- If the total number of concurrent scripts is greater than 1,000, or the number of actuators is configured in **Advanced Config** > **Number of Actuators** of the task, evaluate whether the attributes in the scripts can be used in distributed scenarios (multiple hosts run the scripts at the same time).

- Constant throughput timers are not recommended.

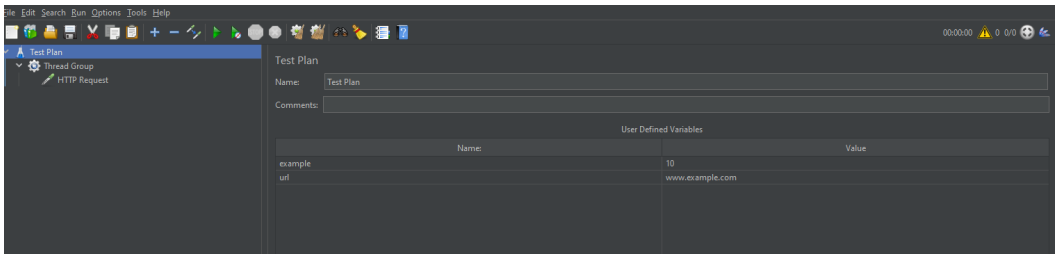
  If the constant throughput timer of JMeter is enabled, the results of pressure test will be inaccurate. You are advised to use the precise throughput timer of JMeter. If the constant throughput timer must be used, evaluate its impact on the actuator performance.

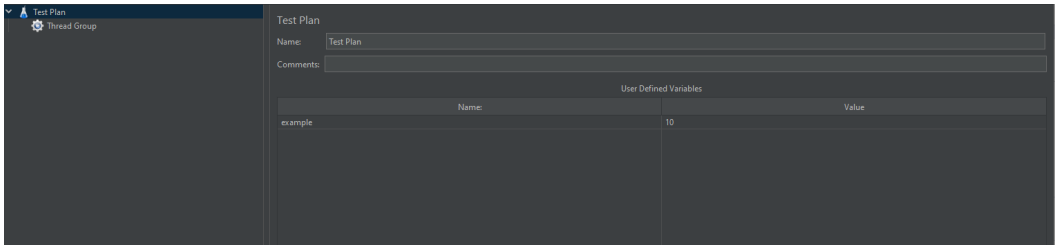
- When you use throughput controllers, the debugging result of the JMeter project may be below expectations.

  The reason is that the CodeArts PerfTest debugging script is executed only once. In this case, you are advised to use concurrent tasks on a small scale in a short time to replace the debugging.

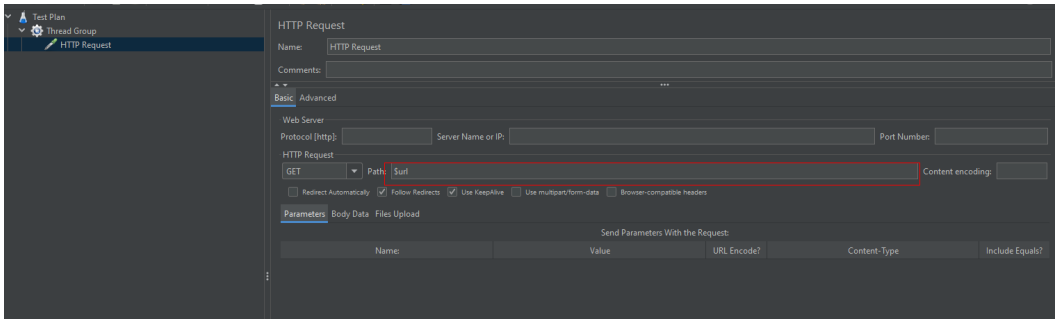
4.5.6 How Do I Use the Global Variable Function?

Step 1 Open the JMeter script, choose **Test Plan**, and define variables in **User Defined Variables** on the right of the page.





Step 2 Reference variables in the script.



Step 3 Log in to the CodeArts PerfTest console. In the navigation pane on the left, choose **JMeter Test Projects**.

Step 4 Click the project whose global variable is to be imported. The **Test Plan List** tab page is displayed.

Step 5 Click the task to which a global variable is to be added. The **Thread Group** page is displayed.

Step 6 Click **Global Variable** in the upper right corner of the page. In the dialog box that is displayed, click **Add Variable**.

Variables are classified into static variables and evenly-split variables.

- Static variable: Content is delivered as a character string. When scripts are executed in distributed mode, the variable values obtained by each node are the same. For example, if the static variable "successRate = 0.8" is delivered, 2,000 tasks are concurrently executed by two actuators, and the value of **successRate** in the script of each actuator is **0.8**.
- Evenly-split variable: Content is delivered as an integer. When scripts are executed in distributed mode, the variable values obtained by each node are evenly distributed. Integer division is used during even distribution, and the remainder is allocated to one of the nodes. For example:
  - If **tps**=**100** and there are four actuators, the value of the variable in the script of each actuator is 25.
  - If **tps**=**20** and there are three actuators, the value of the variable in the script of each actuator is 8, 6, 6.
  - If **tps**=**1** and there are four actuators, the value of the variable in the script of each actuator is 1, 0, 0, 0.

  Pay attention to the following when using the evenly-split variable:

  a. If the allocated value is sensitive to even split, set the number of actuators to ensure that the value of the variable is an integer multiple of the number of actuators.

  b. If the allocated value is not sensitive to even split, increase the allocated value as much as possible to reduce the impact of the integer division on the remainder and evenly split the value.

  c. If the allocated value cannot be 0, set the number of actuators to ensure that the value of the variable is greater than the number of actuators.

NOTE

If a variable configured in the global variable exists in **Test Plan** > **User Defined Variables** of a script, the variable value defined in the script will be overwritten.

Otherwise, the corresponding variable will be created in **Test Plan** > **User Defined Variables** of the script.

----End

4.5.7 What Should I Pay Attention to When Uploading a Third-Party JAR Package?

- Ensure that the third-party JAR package is in the JMeter root directory **/lib/ext** when the local JMeter is working.
- Ensure that the script using the third-party JAR package can run properly on the local host.
- The name (including the extension) of an imported file can contain up to 64 bytes. The maximum file size is 10 MB.

4.5.8 What Should I Pay Attention to When Uploading a CSV File?

When uploading a CSV file to a JMeter project:

- Ensure that parameters in the CSV file can be valued and used in the local JMeter.
- Upload a file in CSV (UTF-8 without BOM) format. Do not upload files in other formats because code reading problems may occur.

4.5.9 What Should I Pay Attention to When Uploading a Custom Installation Package?

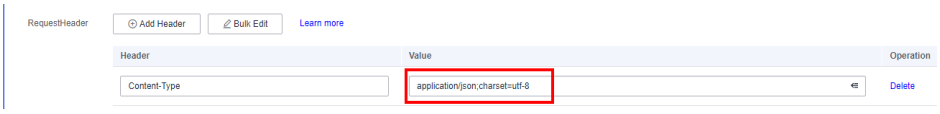
- Be sure to use a ZIP package from the Apache official website or a package with the same directory structure as the Apache ZIP package.
- The JMeter version must be 5.2 to 5.4.
- Third-party plug-ins contained in the ZIP package will affect the JMeter engine of CodeArts PerfTest after being uploaded.

4.5.10 Why Does CodeArts PerfTest Return Garbled Characters When Content-Type in the Request Header Is Set to UTF-8 in JMeter?

When JMeter is used to set **content-type** in the request header to **UTF-8**, the request is returned properly. However, when CodeArts PerfTest is used, garbled characters are returned. This occurs because the UTF-8 encoding format needs to be specified in **content-type** in the request header. Delete undesired request header fields, for example, **Accept-Encoding: gzip**.

Set **content-type** in the request header as follows.

Figure 4-2 Setting of content-type in the request header



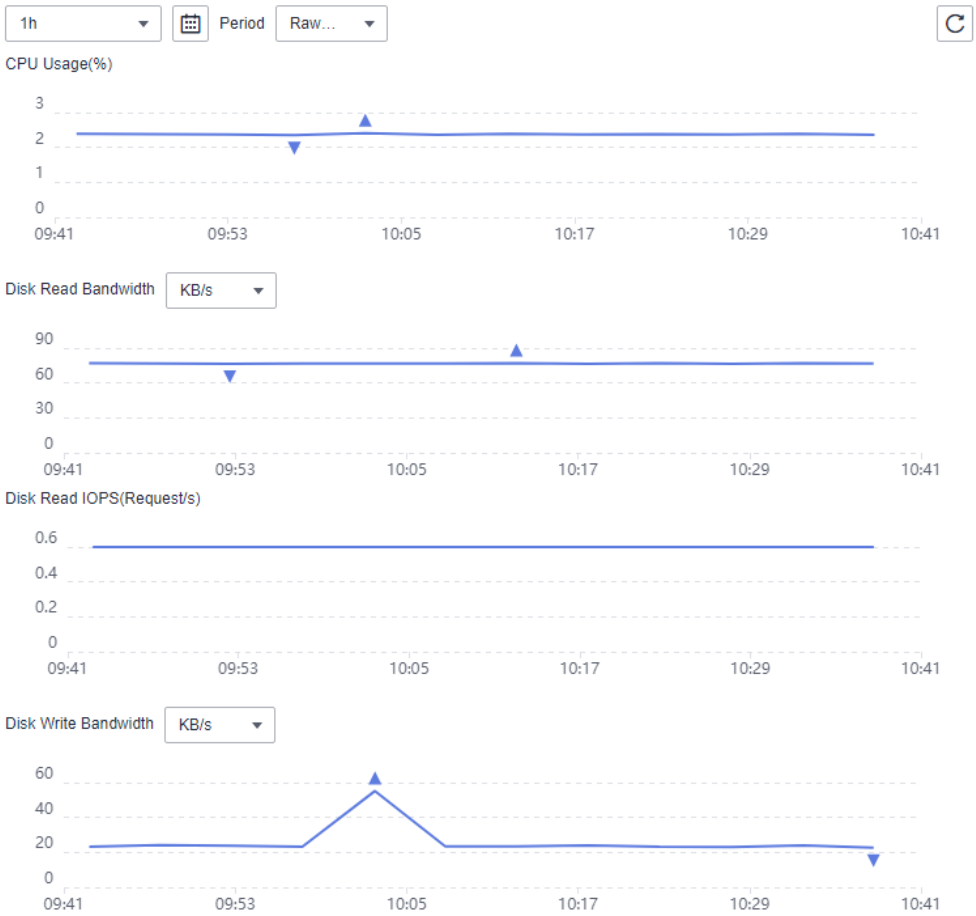
4.5.11 What Are the Meanings of Log Errors in a JMeter Report?

JMeter Timeout in Event Logs

If a JMeter test task does not generate any sampling data in 10 minutes, the task will be forcibly terminated.

"connection Reset" Displayed in Request Logs

It indicates that the network is disconnected. Check the network load of the executor and server. You can check the network bandwidth usage of other cloud services, such as Elastic IP (EIP).



4.5.12 Why Does JMeter Case Debugging Fail in Less Than 5 Seconds and No Data Is Displayed on the Page?

Possible causes:

- Some variable files required by the JMeter test plan have not been uploaded.
- Some third-party JAR packages required by the JMeter test plan have not been uploaded.

Check whether all variable files and third-party JAR packages required have been uploaded.